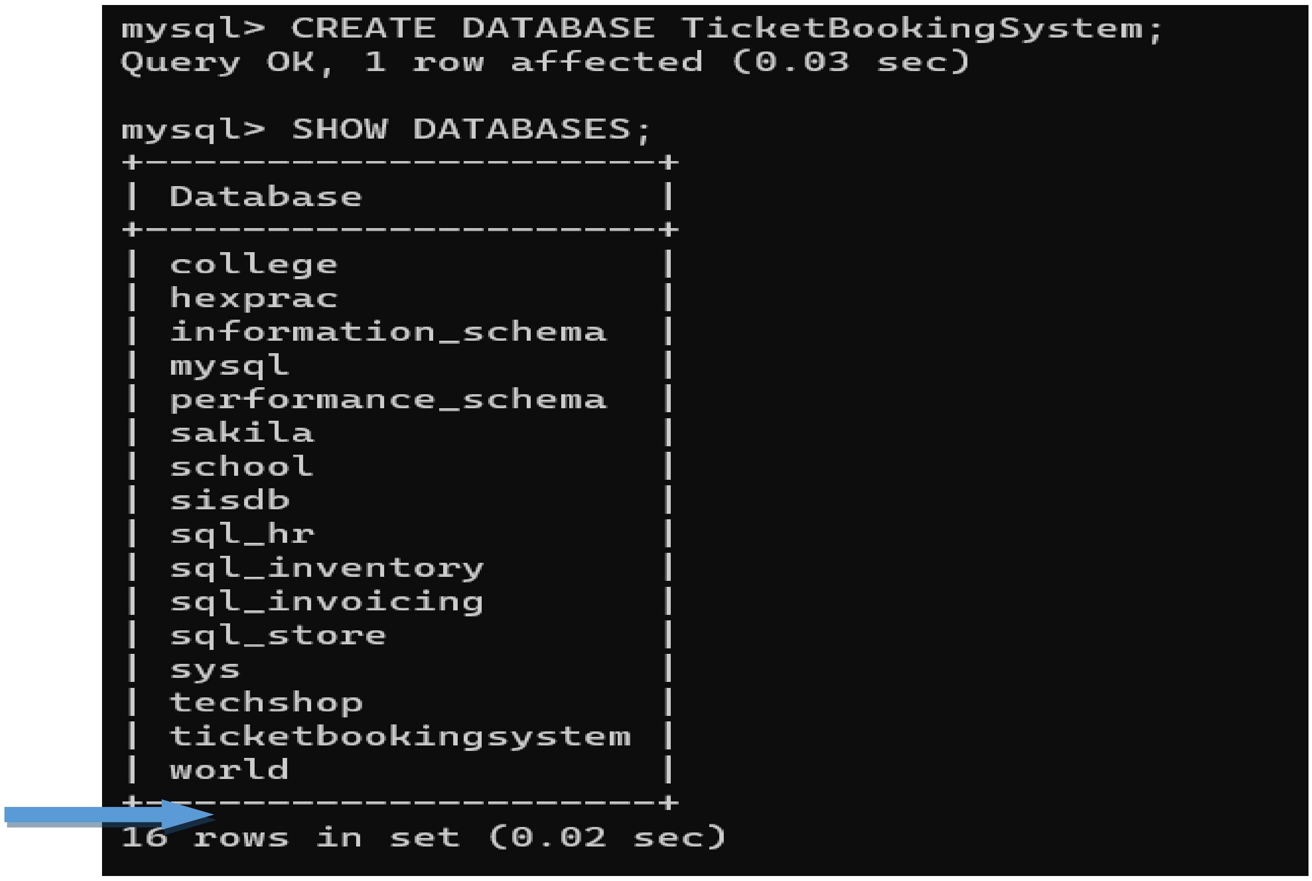
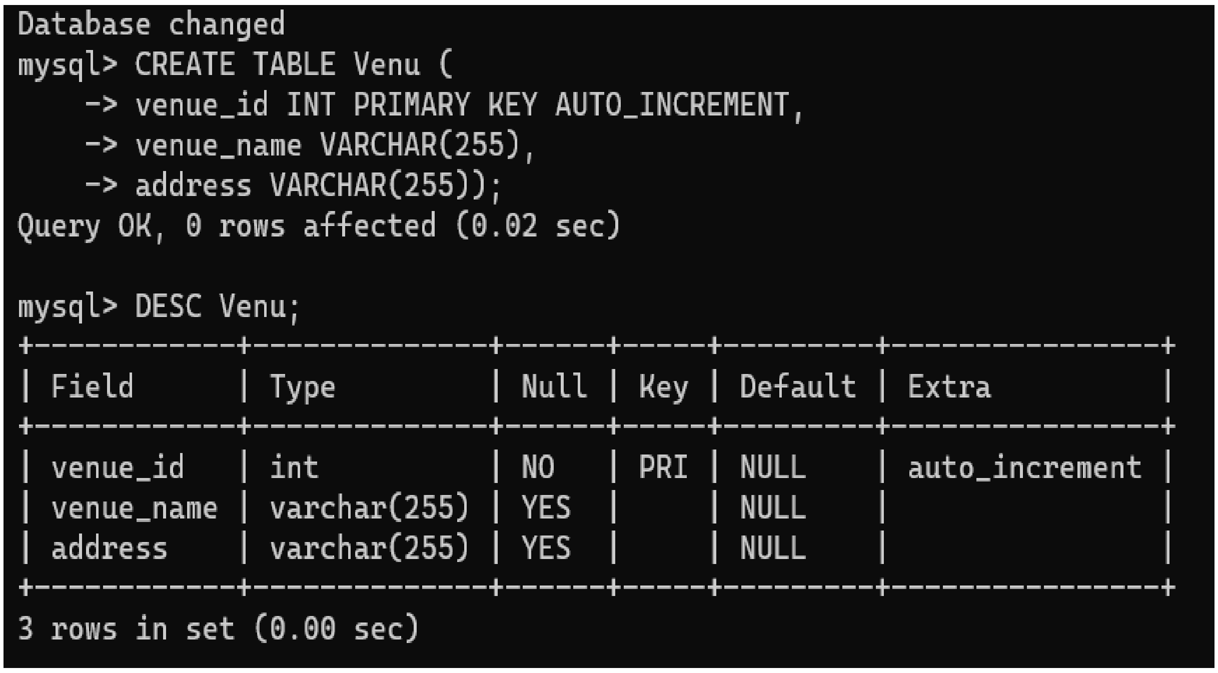
Tasks 1: Database Design

1. Create the database named "TicketBookingSystem".



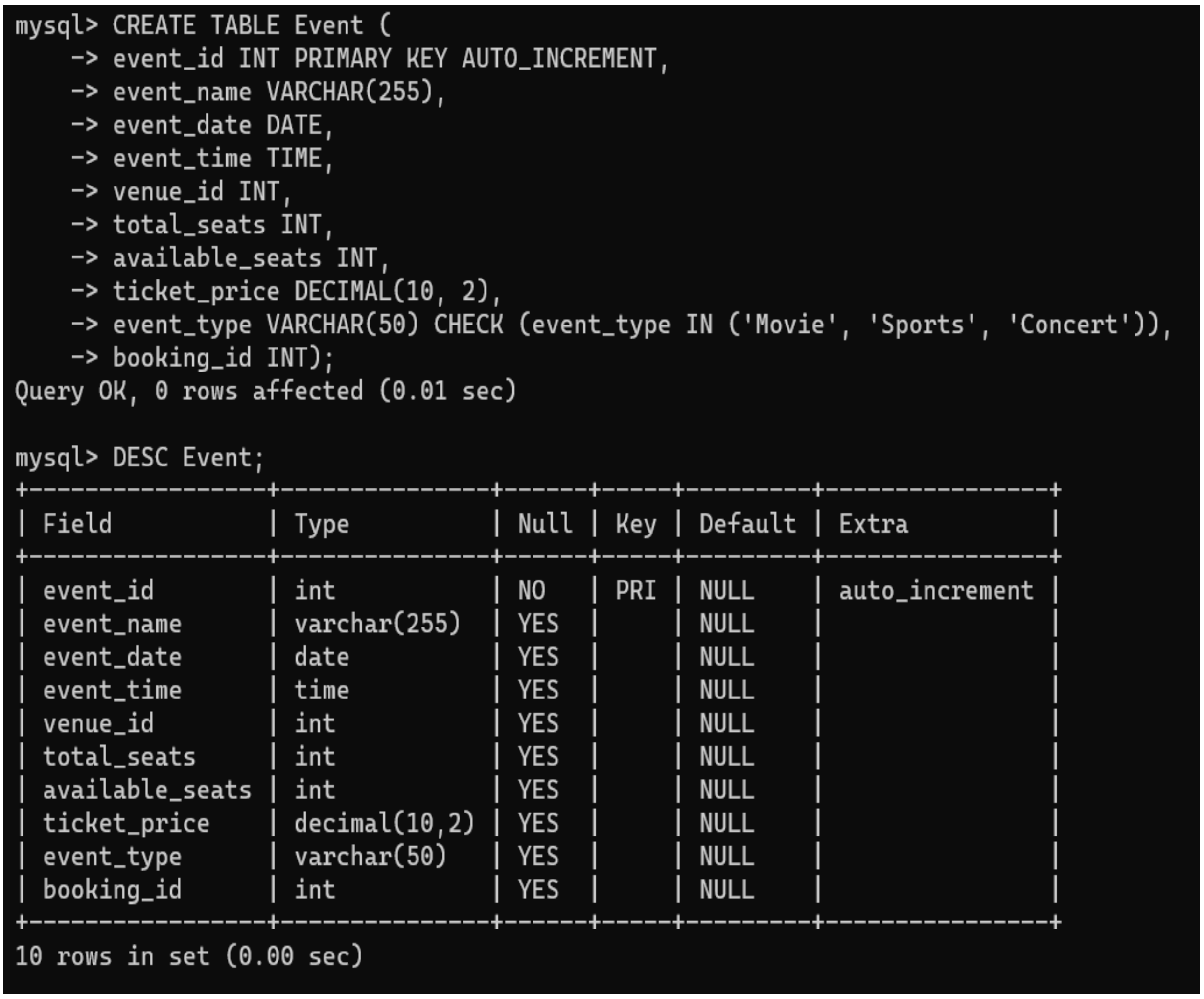
```
mysql> CREATE DATABASE TicketBookingSystem;
Query OK, 1 row affected (0.03 sec)

mysql> SHOW DATABASES;
+----------------------+
| Database             |
+----------------------+
| college              |
| hexprac              |
| information_schema   |
| mysql                |
| performance_schema   |
| sakila               |
| school               |
| sisdb                |
| sql_hr               |
| sql_inventory        |
| sql_invoicing        |
| sql_store            |
| sys                  |
| techshop             |
| ticketbookingsystem  |
| world                |
+----------------------+
16 rows in set (0.02 sec)
```

2. Write SQL scripts to create the mentioned tables with appropriate data types, constraints, and relationships. • Venu • Event • Customers • Booking



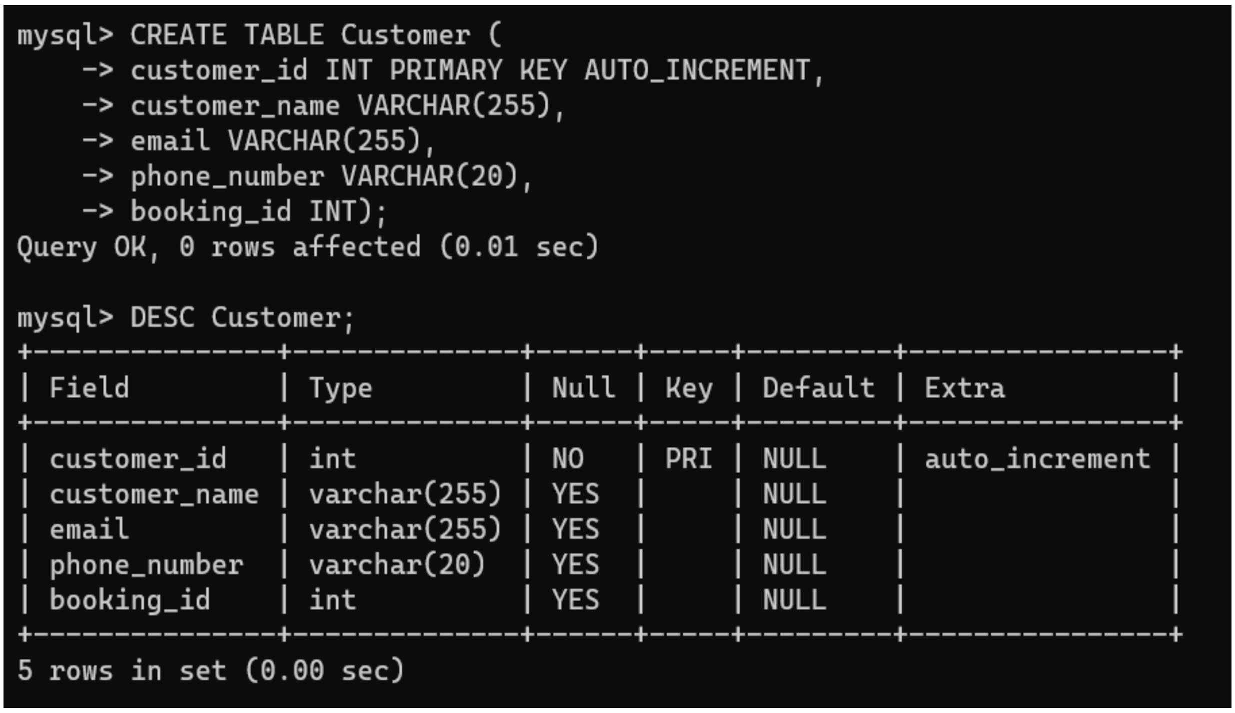
```
Database changed
mysql> CREATE TABLE Venu (
    -> venue_id INT PRIMARY KEY AUTO_INCREMENT,
    -> venue_name VARCHAR(255),
    -> address VARCHAR(255));
Query OK, 0 rows affected (0.02 sec)

mysql> DESC Venu;
+------------+--------------+------+-----+---------+----------------+
| Field      | Type         | Null | Key | Default | Extra          |
+------------+--------------+------+-----+---------+----------------+
| venue_id   | int          | NO   | PRI | NULL    | auto_increment |
| venue_name | varchar(255) | YES  |     | NULL    |                |
| address    | varchar(255) | YES  |     | NULL    |                |
+------------+--------------+------+-----+---------+----------------+
3 rows in set (0.00 sec)
```

(Venu Table)



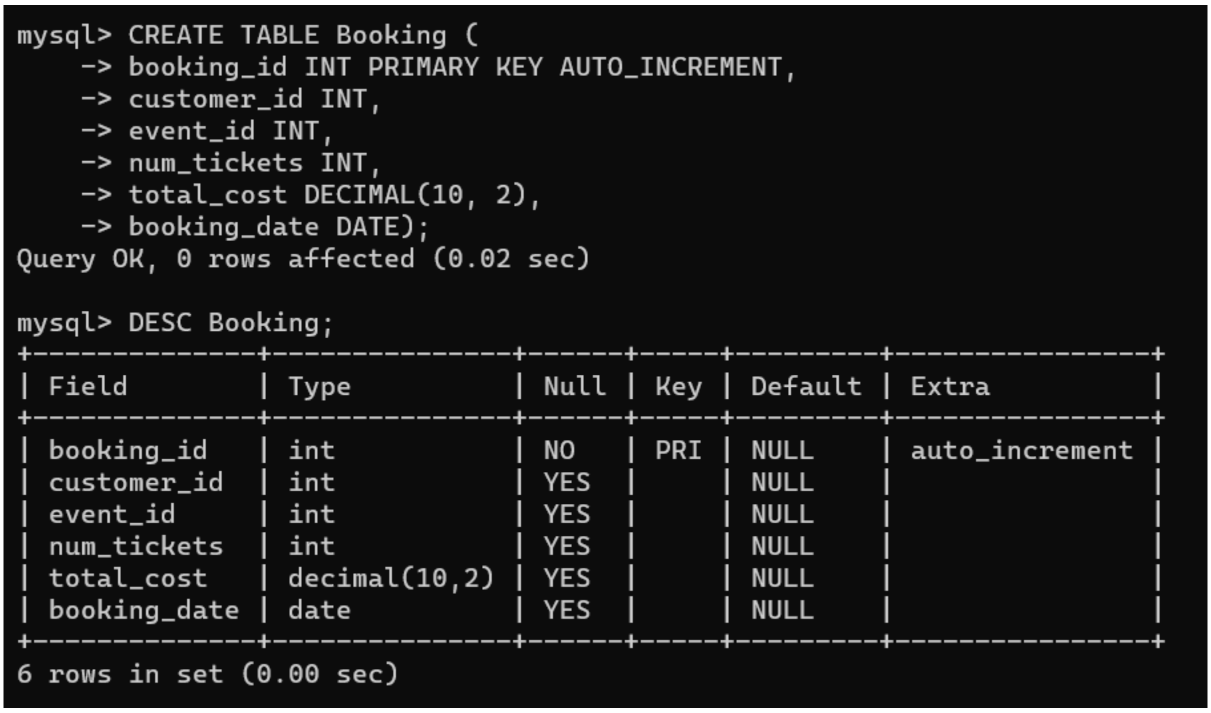
```
mysql> CREATE TABLE Event (
    -> event_id INT PRIMARY KEY AUTO_INCREMENT,
    -> event_name VARCHAR(255),
    -> event_date DATE,
    -> event_time TIME,
    -> venue_id INT,
    -> total_seats INT,
    -> available_seats INT,
    -> ticket_price DECIMAL(10, 2),
    -> event_type VARCHAR(50) CHECK (event_type IN ('Movie', 'Sports', 'Concert')),
    -> booking_id INT);
Query OK, 0 rows affected (0.01 sec)

mysql> DESC Event;
+-----------------+---------------+------+-----+---------+----------------+
| Field           | Type          | Null | Key | Default | Extra          |
+-----------------+---------------+------+-----+---------+----------------+
| event_id        | int           | NO   | PRI | NULL    | auto_increment |
| event_name      | varchar(255)  | YES  |     | NULL    |                |
| event_date      | date          | YES  |     | NULL    |                |
| event_time      | time          | YES  |     | NULL    |                |
| venue_id        | int           | YES  |     | NULL    |                |
| total_seats     | int           | YES  |     | NULL    |                |
| available_seats | int           | YES  |     | NULL    |                |
| ticket_price    | decimal(10,2) | YES  |     | NULL    |                |
| event_type      | varchar(50)   | YES  |     | NULL    |                |
| booking_id      | int           | YES  |     | NULL    |                |
+-----------------+---------------+------+-----+---------+----------------+
10 rows in set (0.00 sec)
```

(Event Table)



```
mysql> CREATE TABLE Customer (
    -> customer_id INT PRIMARY KEY AUTO_INCREMENT,
    -> customer_name VARCHAR(255),
    -> email VARCHAR(255),
    -> phone_number VARCHAR(20),
    -> booking_id INT);
Query OK, 0 rows affected (0.01 sec)

mysql> DESC Customer;
+---------------+--------------+------+-----+---------+----------------+
| Field         | Type         | Null | Key | Default | Extra          |
+---------------+--------------+------+-----+---------+----------------+
| customer_id   | int          | NO   | PRI | NULL    | auto_increment |
| customer_name | varchar(255) | YES  |     | NULL    |                |
| email         | varchar(255) | YES  |     | NULL    |                |
| phone_number  | varchar(20)  | YES  |     | NULL    |                |
| booking_id    | int          | YES  |     | NULL    |                |
+---------------+--------------+------+-----+---------+----------------+
5 rows in set (0.00 sec)
```

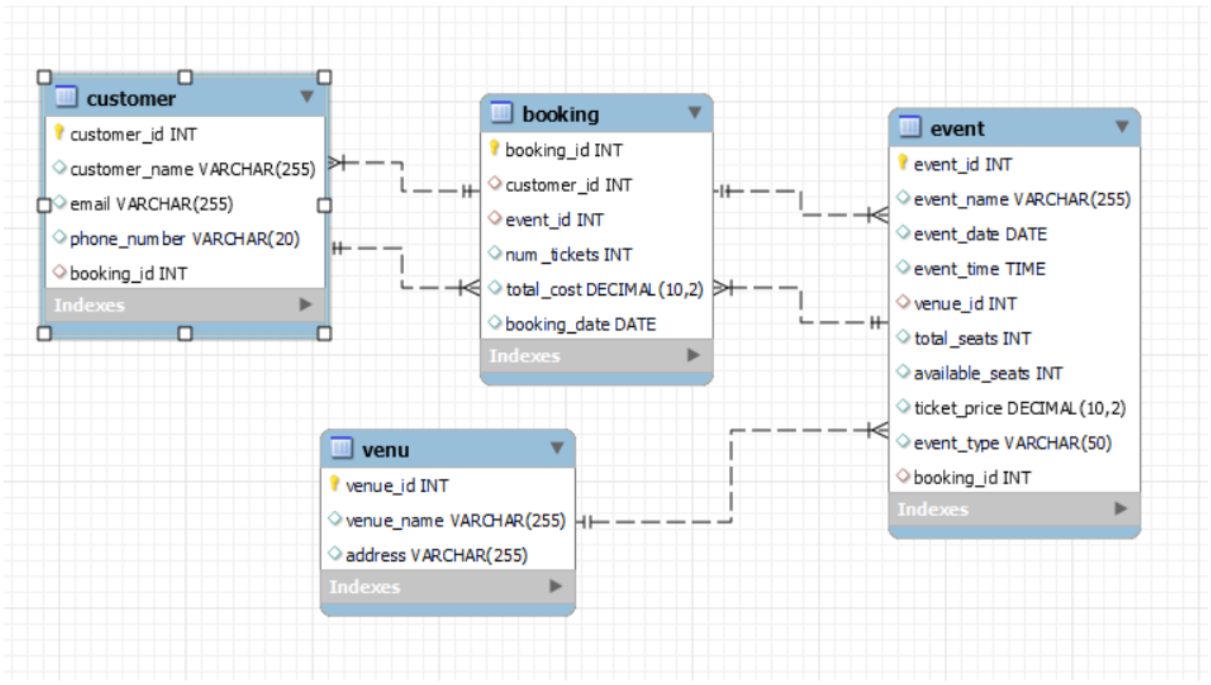
(Customer Table)



```
mysql> CREATE TABLE Booking (
    -> booking_id INT PRIMARY KEY AUTO_INCREMENT,
    -> customer_id INT,
    -> event_id INT,
    -> num_tickets INT,
    -> total_cost DECIMAL(10, 2),
    -> booking_date DATE);
Query OK, 0 rows affected (0.02 sec)

mysql> DESC Booking;
+--------------+---------------+------+-----+---------+----------------+
| Field        | Type          | Null | Key | Default | Extra          |
+--------------+---------------+------+-----+---------+----------------+
| booking_id   | int           | NO   | PRI | NULL    | auto_increment |
| customer_id  | int           | YES  |     | NULL    |                |
| event_id     | int           | YES  |     | NULL    |                |
| num_tickets  | int           | YES  |     | NULL    |                |
| total_cost   | decimal(10,2) | YES  |     | NULL    |                |
| booking_date | date          | YES  |     | NULL    |                |
+--------------+---------------+------+-----+---------+----------------+
6 rows in set (0.00 sec)
```

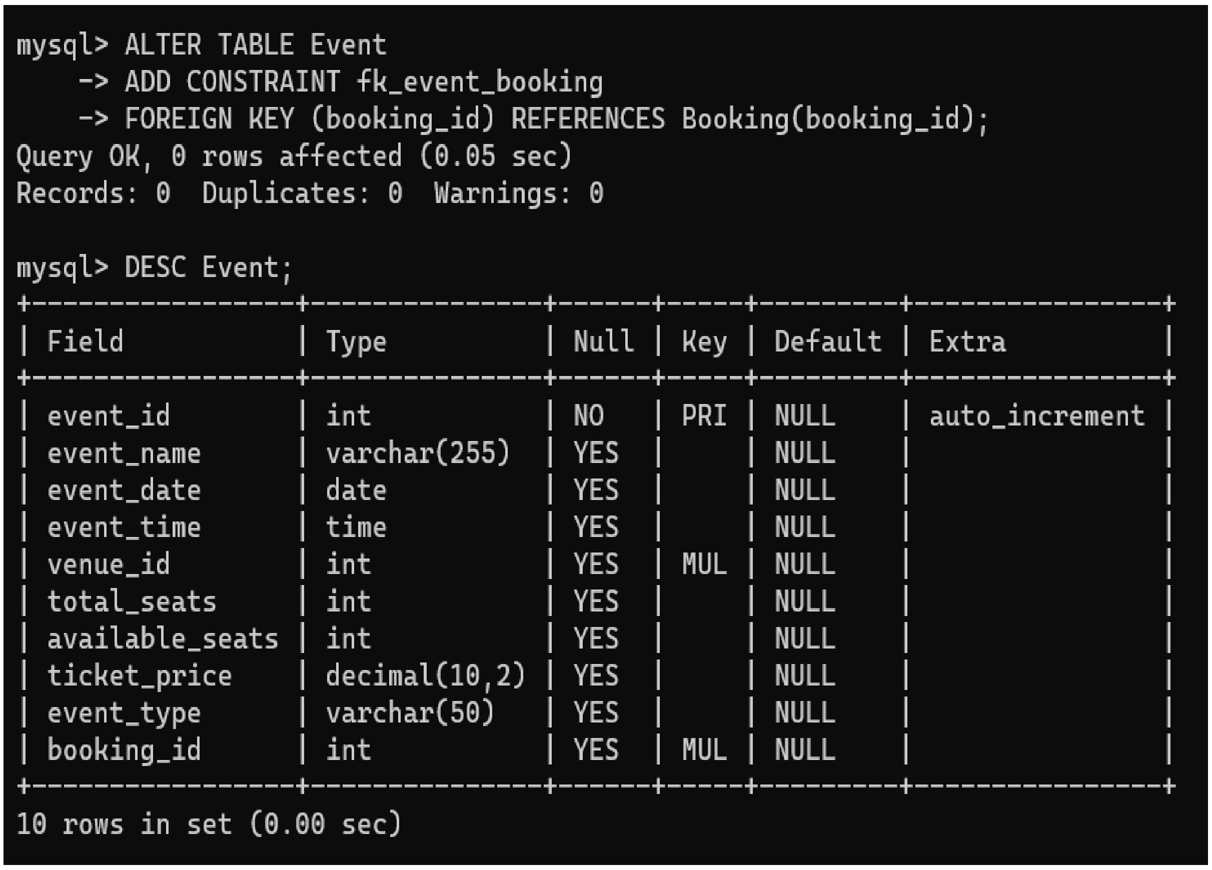
(Booking Table)

3. Create an ERD (Entity Relationship Diagram) for the database.



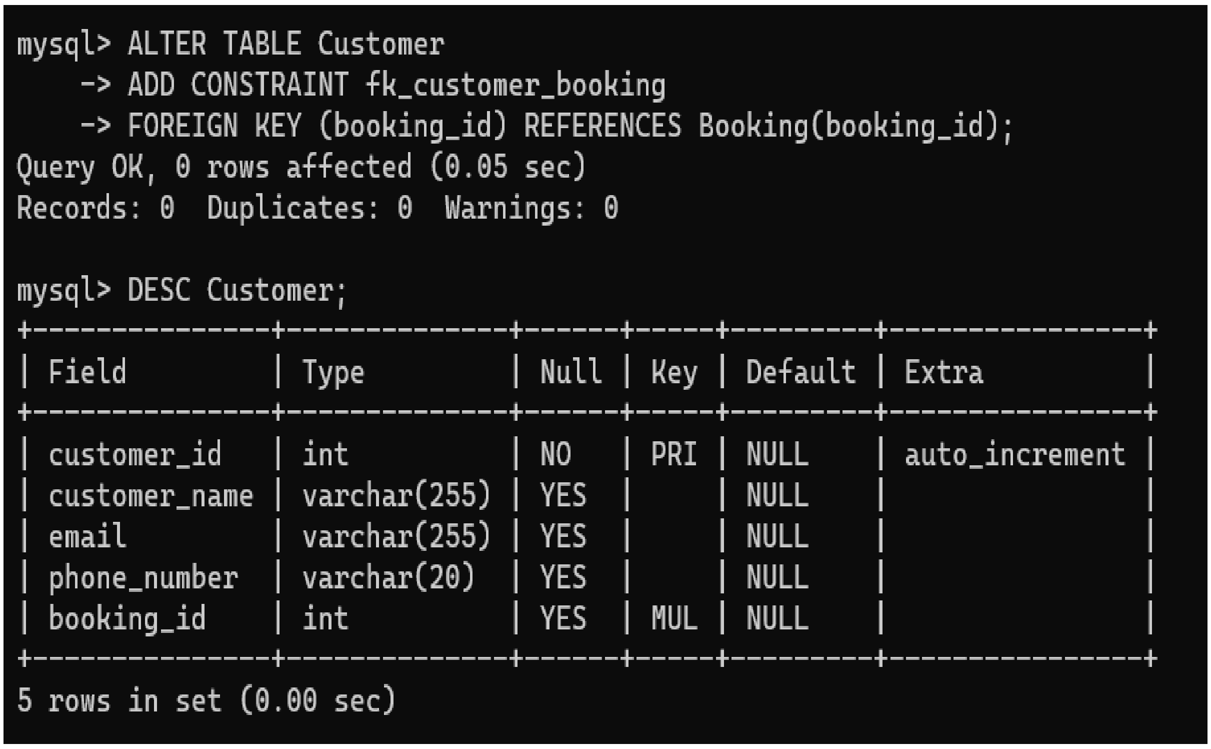
(ERD for TicketBookingSystem)

4. Create appropriate Primary Key and Foreign Key constraints for referential integrity.



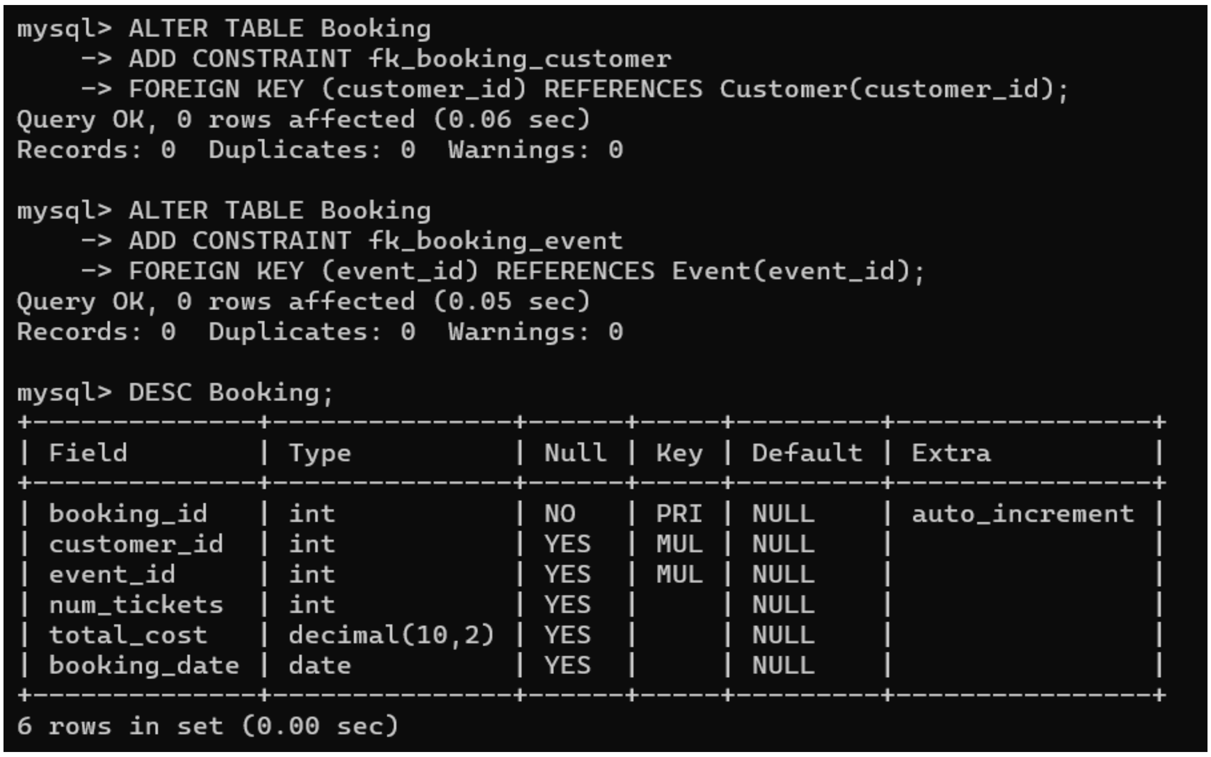
```
mysql> ALTER TABLE Event
    -> ADD CONSTRAINT fk_event_booking
    -> FOREIGN KEY (booking_id) REFERENCES Booking(booking_id);
Query OK, 0 rows affected (0.05 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> DESC Event;
+-----------------+---------------+------+-----+---------+----------------+
| Field           | Type          | Null | Key | Default | Extra          |
+-----------------+---------------+------+-----+---------+----------------+
| event_id        | int           | NO   | PRI | NULL    | auto_increment |
| event_name      | varchar(255)  | YES  |     | NULL    |                |
| event_date      | date          | YES  |     | NULL    |                |
| event_time      | time          | YES  |     | NULL    |                |
| venue_id        | int           | YES  | MUL | NULL    |                |
| total_seats     | int           | YES  |     | NULL    |                |
| available_seats | int           | YES  |     | NULL    |                |
| ticket_price    | decimal(10,2) | YES  |     | NULL    |                |
| event_type      | varchar(50)   | YES  |     | NULL    |                |
| booking_id      | int           | YES  | MUL | NULL    |                |
+-----------------+---------------+------+-----+---------+----------------+
10 rows in set (0.00 sec)
```

(Event Table)



```
mysql> ALTER TABLE Customer
    -> ADD CONSTRAINT fk_customer_booking
    -> FOREIGN KEY (booking_id) REFERENCES Booking(booking_id);
Query OK, 0 rows affected (0.05 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> DESC Customer;
+---------------+--------------+------+-----+---------+----------------+
| Field         | Type         | Null | Key | Default | Extra          |
+---------------+--------------+------+-----+---------+----------------+
| customer_id   | int          | NO   | PRI | NULL    | auto_increment |
| customer_name | varchar(255) | YES  |     | NULL    |                |
| email         | varchar(255) | YES  |     | NULL    |                |
| phone_number  | varchar(20)  | YES  |     | NULL    |                |
| booking_id    | int          | YES  | MUL | NULL    |                |
+---------------+--------------+------+-----+---------+----------------+
5 rows in set (0.00 sec)
```

(Customer Table)

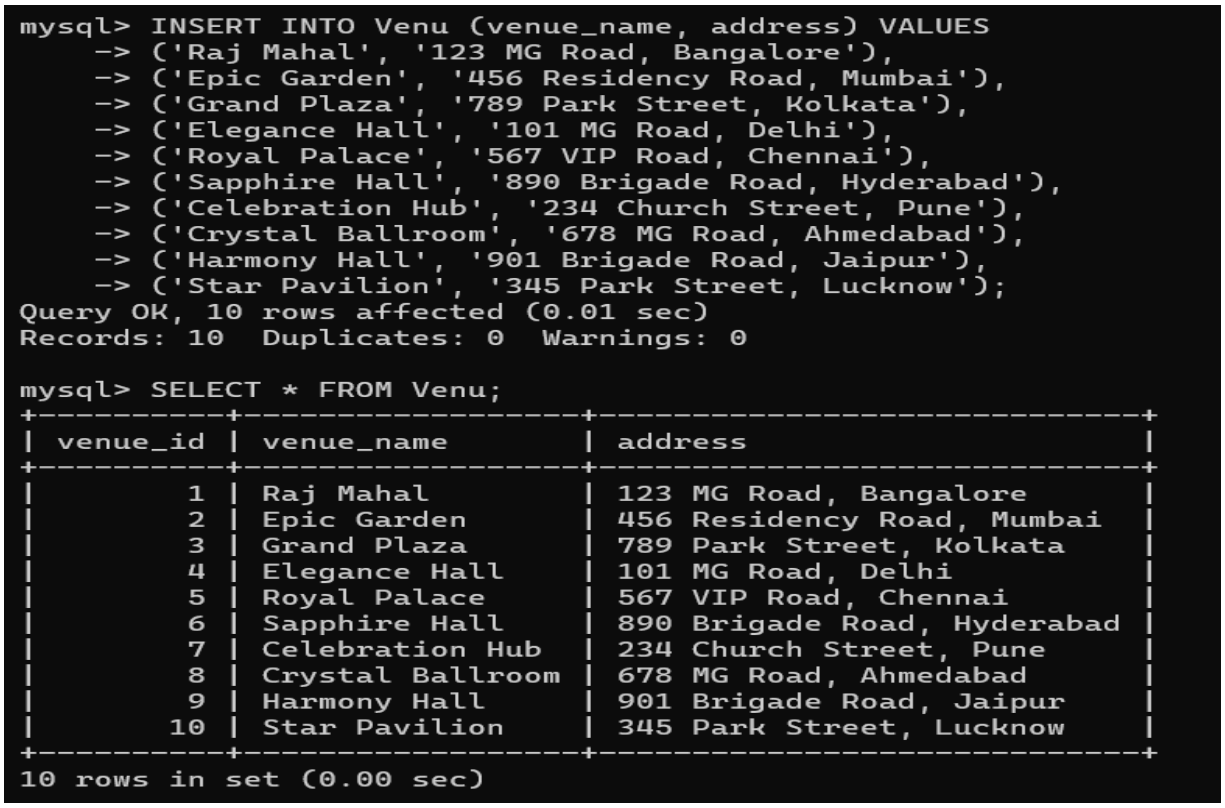


```
mysql> ALTER TABLE Booking
    -> ADD CONSTRAINT fk_booking_customer
    -> FOREIGN KEY (customer_id) REFERENCES Customer(customer_id);
Query OK, 0 rows affected (0.06 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> ALTER TABLE Booking
    -> ADD CONSTRAINT fk_booking_event
    -> FOREIGN KEY (event_id) REFERENCES Event(event_id);
Query OK, 0 rows affected (0.05 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> DESC Booking;
+--------------+---------------+------+-----+---------+----------------+
| Field        | Type          | Null | Key | Default | Extra          |
+--------------+---------------+------+-----+---------+----------------+
| booking_id   | int           | NO   | PRI | NULL    | auto_increment |
| customer_id  | int           | YES  | MUL | NULL    |                |
| event_id     | int           | YES  | MUL | NULL    |                |
| num_tickets  | int           | YES  |     | NULL    |                |
| total_cost   | decimal(10,2) | YES  |     | NULL    |                |
| booking_date | date          | YES  |     | NULL    |                |
+--------------+---------------+------+-----+---------+----------------+
6 rows in set (0.00 sec)
```

(Booking Table)

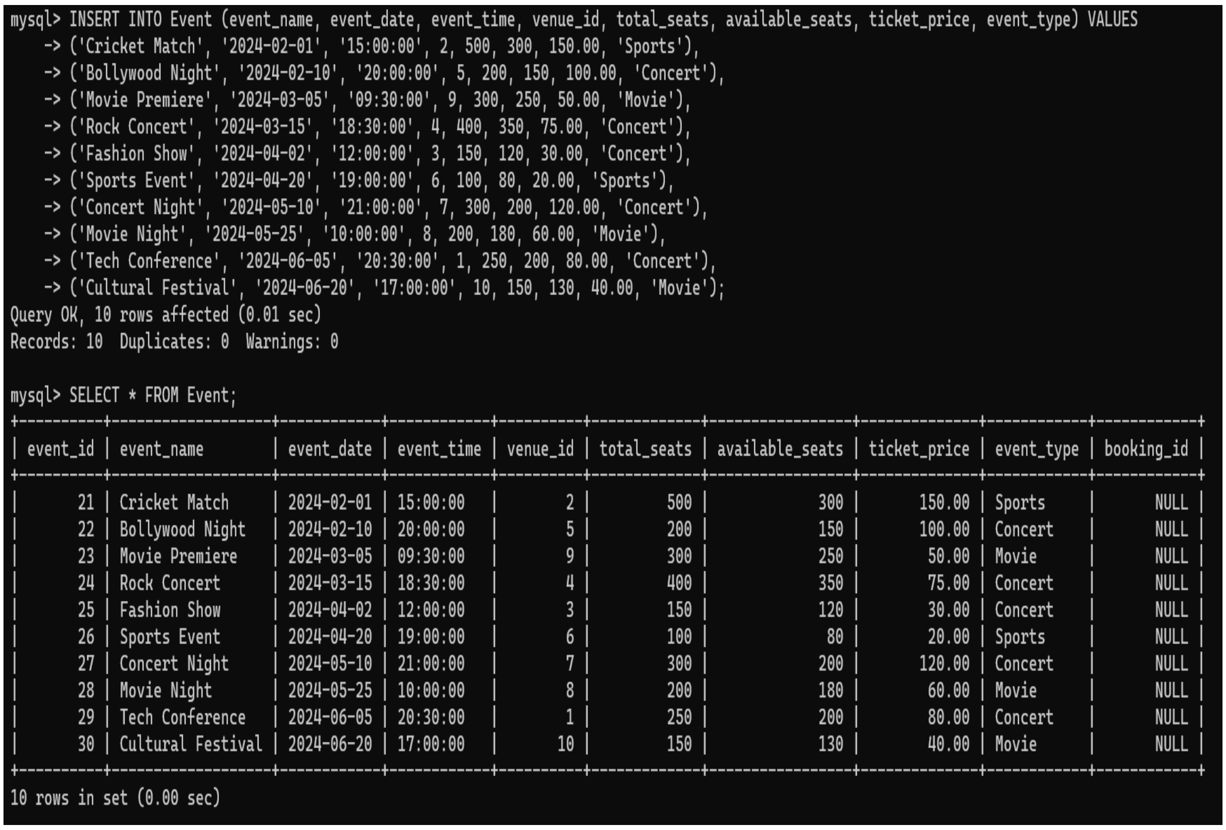
Tasks 2: Select, Where, Between, AND, LIKE

1. Write a SQL query to insert at least 10 sample records into each table.



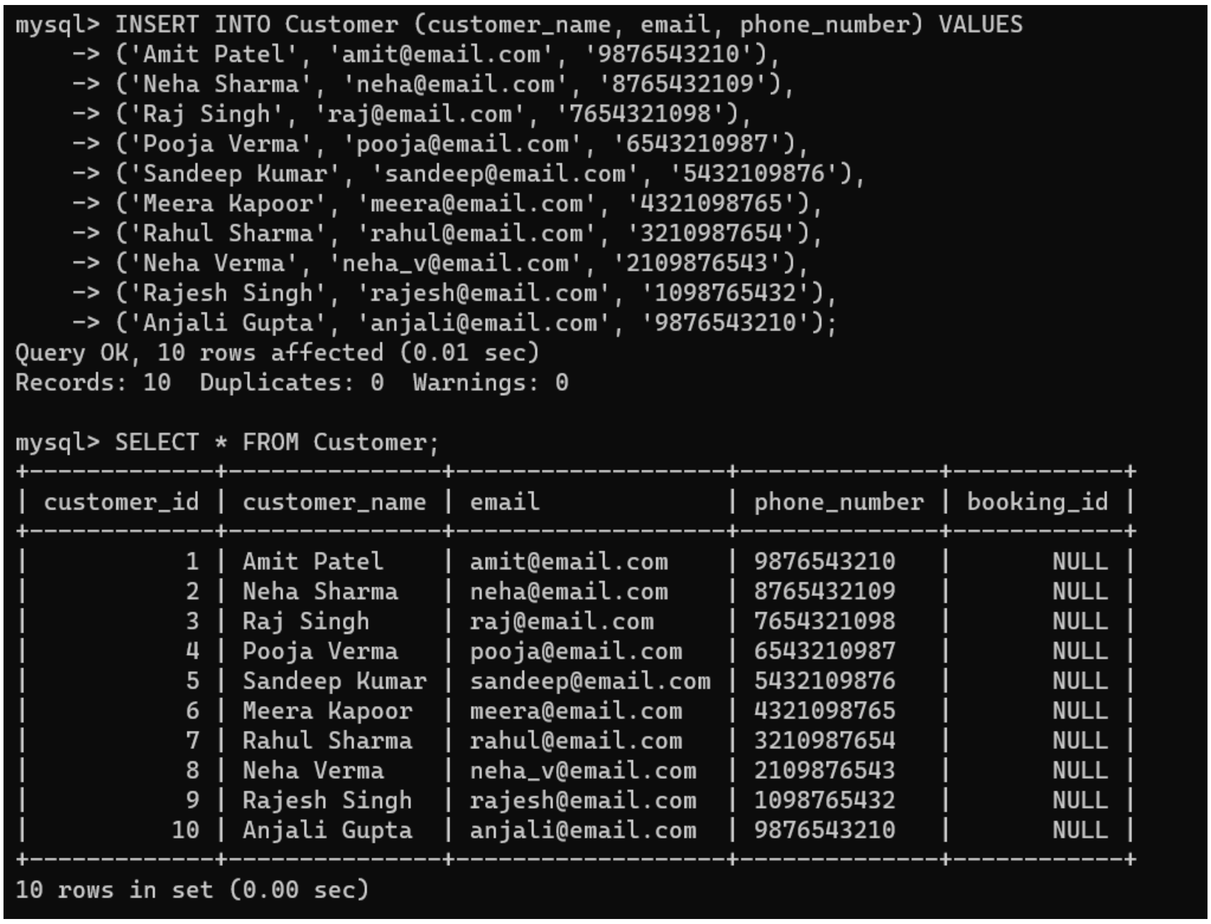
```
mysql> INSERT INTO Venu (venue_name, address) VALUES
    -> ('Raj Mahal', '123 MG Road, Bangalore'),
    -> ('Epic Garden', '456 Residency Road, Mumbai'),
    -> ('Grand Plaza', '789 Park Street, Kolkata'),
    -> ('Elegance Hall', '101 MG Road, Delhi'),
    -> ('Royal Palace', '567 VIP Road, Chennai'),
    -> ('Sapphire Hall', '890 Brigade Road, Hyderabad'),
    -> ('Celebration Hub', '234 Church Street, Pune'),
    -> ('Crystal Ballroom', '678 MG Road, Ahmedabad'),
    -> ('Harmony Hall', '901 Brigade Road, Jaipur'),
    -> ('Star Pavilion', '345 Park Street, Lucknow');
Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM Venu;
+----------+------------------+-----------------------------+
| venue_id | venue_name       | address                     |
+----------+------------------+-----------------------------+
|        1 | Raj Mahal        | 123 MG Road, Bangalore      |
|        2 | Epic Garden      | 456 Residency Road, Mumbai  |
|        3 | Grand Plaza      | 789 Park Street, Kolkata    |
|        4 | Elegance Hall    | 101 MG Road, Delhi          |
|        5 | Royal Palace     | 567 VIP Road, Chennai       |
|        6 | Sapphire Hall    | 890 Brigade Road, Hyderabad |
|        7 | Celebration Hub  | 234 Church Street, Pune     |
|        8 | Crystal Ballroom | 678 MG Road, Ahmedabad      |
|        9 | Harmony Hall     | 901 Brigade Road, Jaipur    |
|       10 | Star Pavilion    | 345 Park Street, Lucknow    |
+----------+------------------+-----------------------------+
10 rows in set (0.00 sec)
```

(Venu Table)



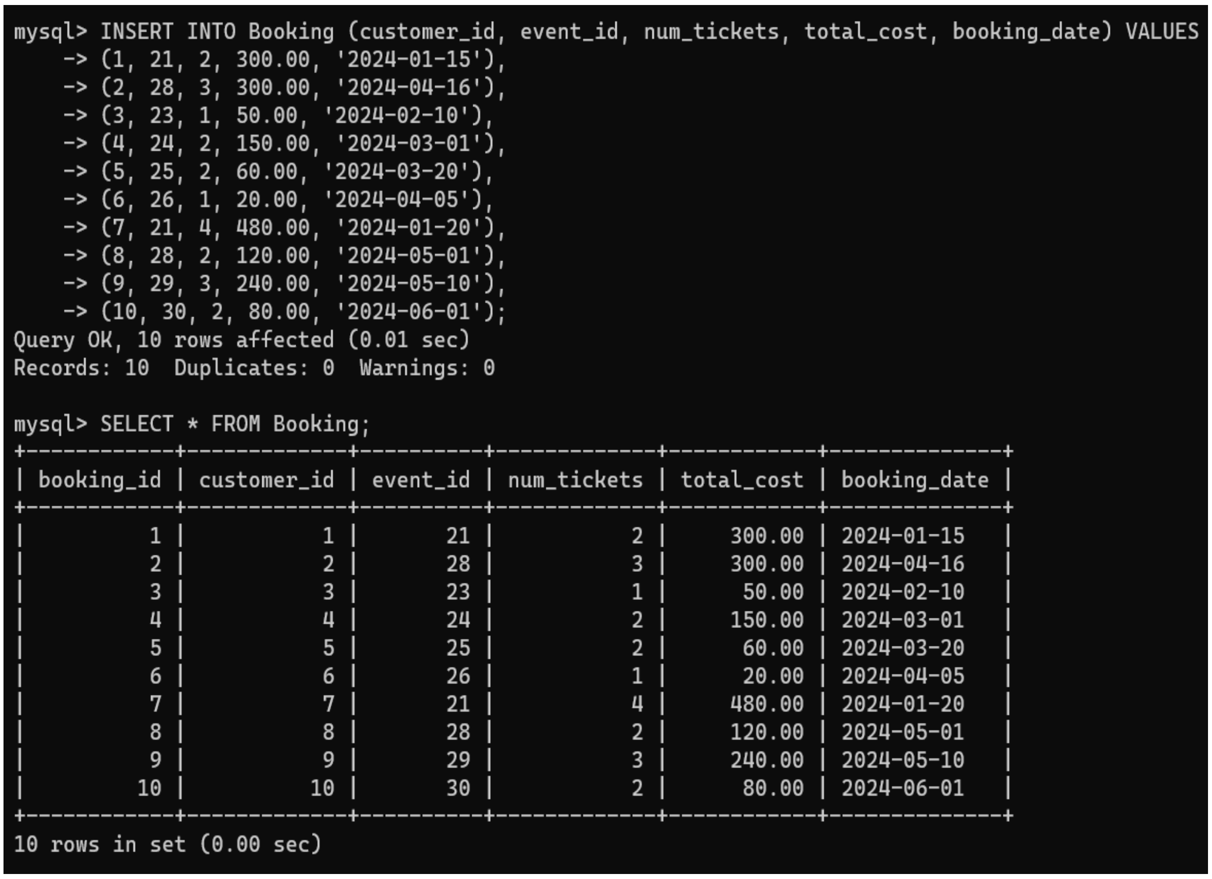
```
mysql> INSERT INTO Event (event_name, event_date, event_time, venue_id, total_seats, available_seats, ticket_price, event_type) VALUES
    -> ('Cricket Match', '2024-02-01', '15:00:00', 2, 500, 300, 150.00, 'Sports'),
    -> ('Bollywood Night', '2024-02-10', '20:00:00', 5, 200, 150, 100.00, 'Concert'),
    -> ('Movie Premiere', '2024-03-05', '09:30:00', 9, 300, 250, 50.00, 'Movie'),
    -> ('Rock Concert', '2024-03-15', '18:30:00', 4, 400, 350, 75.00, 'Concert'),
    -> ('Fashion Show', '2024-04-02', '12:00:00', 3, 150, 120, 30.00, 'Concert'),
    -> ('Sports Event', '2024-04-20', '19:00:00', 6, 100, 80, 20.00, 'Sports'),
    -> ('Concert Night', '2024-05-10', '21:00:00', 7, 300, 200, 120.00, 'Concert'),
    -> ('Movie Night', '2024-05-25', '10:00:00', 8, 200, 180, 60.00, 'Movie'),
    -> ('Tech Conference', '2024-06-05', '20:30:00', 1, 250, 200, 80.00, 'Concert'),
    -> ('Cultural Festival', '2024-06-20', '17:00:00', 10, 150, 130, 40.00, 'Movie');
Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM Event;
+----------+-------------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
| event_id | event_name        | event_date | event_time | venue_id | total_seats | available_seats | ticket_price | event_type | booking_id |
+----------+-------------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
|       21 | Cricket Match     | 2024-02-01 | 15:00:00   |        2 |         500 |             300 |       150.00 | Sports     |       NULL |
|       22 | Bollywood Night   | 2024-02-10 | 20:00:00   |        5 |         200 |             150 |       100.00 | Concert    |       NULL |
|       23 | Movie Premiere    | 2024-03-05 | 09:30:00   |        9 |         300 |             250 |        50.00 | Movie      |       NULL |
|       24 | Rock Concert      | 2024-03-15 | 18:30:00   |        4 |         400 |             350 |        75.00 | Concert    |       NULL |
|       25 | Fashion Show      | 2024-04-02 | 12:00:00   |        3 |         150 |             120 |        30.00 | Concert    |       NULL |
|       26 | Sports Event      | 2024-04-20 | 19:00:00   |        6 |         100 |              80 |        20.00 | Sports     |       NULL |
|       27 | Concert Night     | 2024-05-10 | 21:00:00   |        7 |         300 |             200 |       120.00 | Concert    |       NULL |
|       28 | Movie Night       | 2024-05-25 | 10:00:00   |        8 |         200 |             180 |        60.00 | Movie      |       NULL |
|       29 | Tech Conference   | 2024-06-05 | 20:30:00   |        1 |         250 |             200 |        80.00 | Concert    |       NULL |
|       30 | Cultural Festival | 2024-06-20 | 17:00:00   |       10 |         150 |             130 |        40.00 | Movie      |       NULL |
+----------+-------------------+------------+------------+----------+-------------+-----------------+--------------+------------+------------+
10 rows in set (0.00 sec)
```

(Event Table)



```
mysql> INSERT INTO Customer (customer_name, email, phone_number) VALUES
    -> ('Amit Patel', 'amit@email.com', '9876543210'),
    -> ('Neha Sharma', 'neha@email.com', '8765432109'),
    -> ('Raj Singh', 'raj@email.com', '7654321098'),
    -> ('Pooja Verma', 'pooja@email.com', '6543210987'),
    -> ('Sandeep Kumar', 'sandeep@email.com', '5432109876'),
    -> ('Meera Kapoor', 'meera@email.com', '4321098765'),
    -> ('Rahul Sharma', 'rahul@email.com', '3210987654'),
    -> ('Neha Verma', 'neha_v@email.com', '2109876543'),
    -> ('Rajesh Singh', 'rajesh@email.com', '1098765432'),
    -> ('Anjali Gupta', 'anjali@email.com', '9876543210');
Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM Customer;
+-------------+---------------+-------------------+--------------+------------+
| customer_id | customer_name | email             | phone_number | booking_id |
+-------------+---------------+-------------------+--------------+------------+
|           1 | Amit Patel    | amit@email.com    | 9876543210   |       NULL |
|           2 | Neha Sharma   | neha@email.com    | 8765432109   |       NULL |
|           3 | Raj Singh     | raj@email.com     | 7654321098   |       NULL |
|           4 | Pooja Verma   | pooja@email.com   | 6543210987   |       NULL |
|           5 | Sandeep Kumar | sandeep@email.com | 5432109876   |       NULL |
|           6 | Meera Kapoor  | meera@email.com   | 4321098765   |       NULL |
|           7 | Rahul Sharma  | rahul@email.com   | 3210987654   |       NULL |
|           8 | Neha Verma    | neha_v@email.com  | 2109876543   |       NULL |
|           9 | Rajesh Singh  | rajesh@email.com  | 1098765432   |       NULL |
|          10 | Anjali Gupta  | anjali@email.com  | 9876543210   |       NULL |
+-------------+---------------+-------------------+--------------+------------+
10 rows in set (0.00 sec)
```

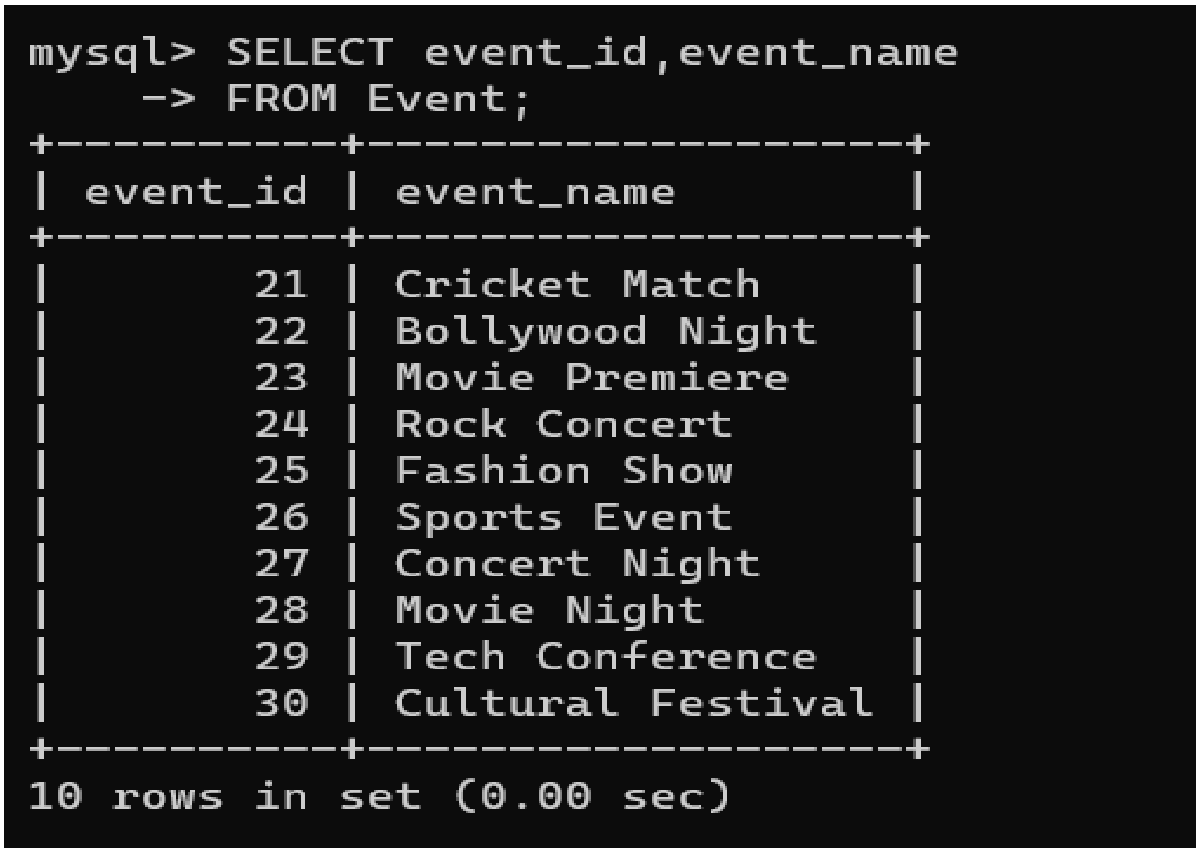
(Customer Table)



```
mysql> INSERT INTO Booking (customer_id, event_id, num_tickets, total_cost, booking_date) VALUES
    -> (1, 21, 2, 300.00, '2024-01-15'),
    -> (2, 28, 3, 300.00, '2024-04-16'),
    -> (3, 23, 1, 50.00, '2024-02-10'),
    -> (4, 24, 2, 150.00, '2024-03-01'),
    -> (5, 25, 2, 60.00, '2024-03-20'),
    -> (6, 26, 1, 20.00, '2024-04-05'),
    -> (7, 21, 4, 480.00, '2024-01-20'),
    -> (8, 28, 2, 120.00, '2024-05-01'),
    -> (9, 29, 3, 240.00, '2024-05-10'),
    -> (10, 30, 2, 80.00, '2024-06-01');
Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM Booking;
+------------+-------------+----------+-------------+------------+--------------+
| booking_id | customer_id | event_id | num_tickets | total_cost | booking_date |
+------------+-------------+----------+-------------+------------+--------------+
|          1 |           1 |       21 |           2 |     300.00 | 2024-01-15   |
|          2 |           2 |       28 |           3 |     300.00 | 2024-04-16   |
|          3 |           3 |       23 |           1 |      50.00 | 2024-02-10   |
|          4 |           4 |       24 |           2 |     150.00 | 2024-03-01   |
|          5 |           5 |       25 |           2 |      60.00 | 2024-03-20   |
|          6 |           6 |       26 |           1 |      20.00 | 2024-04-05   |
|          7 |           7 |       21 |           4 |     480.00 | 2024-01-20   |
|          8 |           8 |       28 |           2 |     120.00 | 2024-05-01   |
|          9 |           9 |       29 |           3 |     240.00 | 2024-05-10   |
|         10 |          10 |       30 |           2 |      80.00 | 2024-06-01   |
+------------+-------------+----------+-------------+------------+--------------+
10 rows in set (0.00 sec)
```

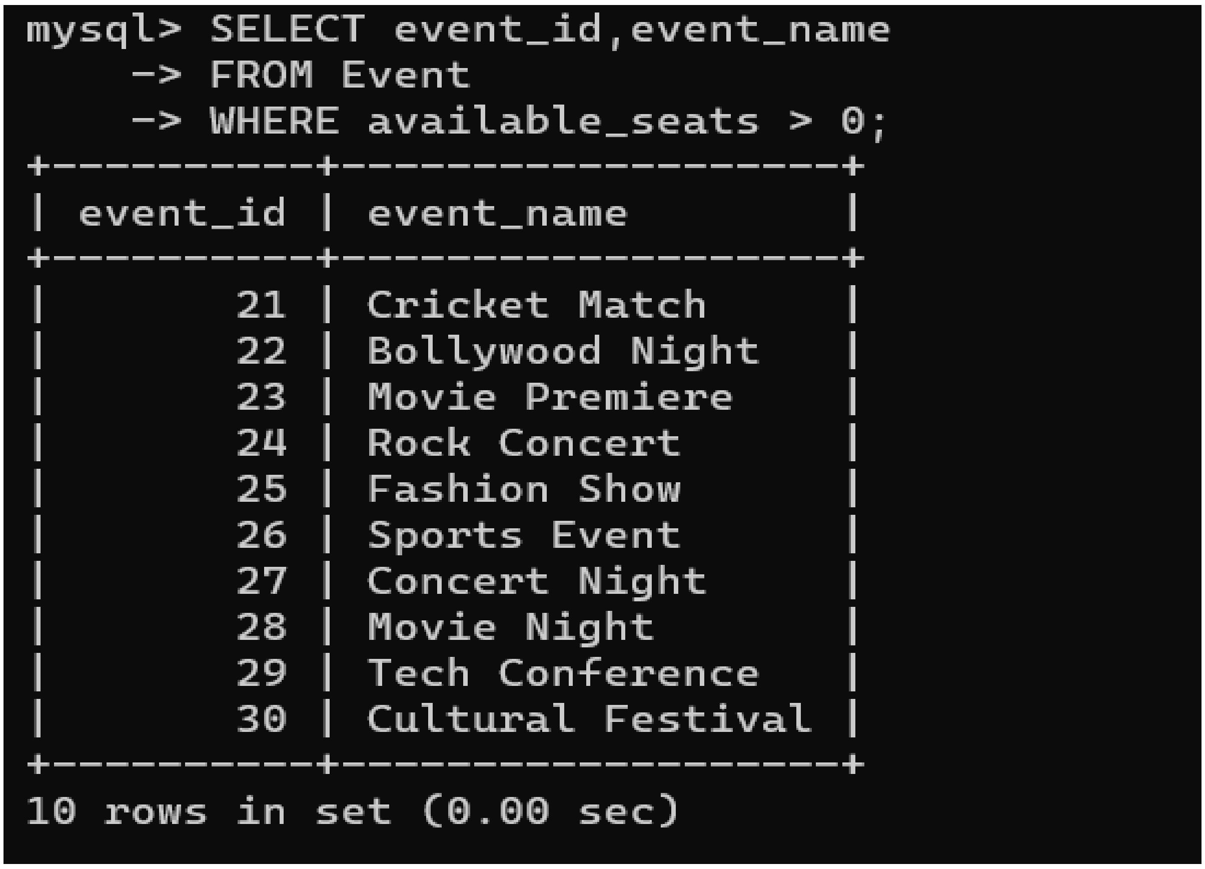
(Booking Table)

2. Write a SQL query to list all Events.



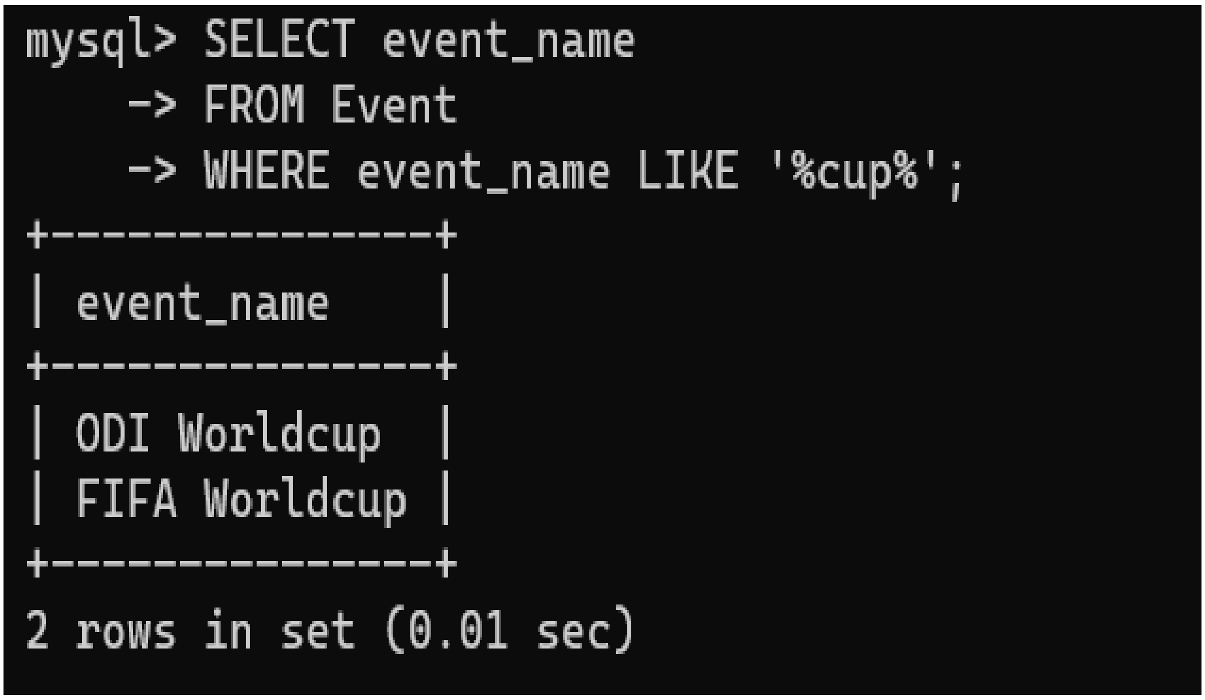
```
mysql> SELECT event_id,event_name
    -> FROM Event;
+----------+--------------------+
| event_id | event_name         |
+----------+--------------------+
|       21 | Cricket Match      |
|       22 | Bollywood Night    |
|       23 | Movie Premiere     |
|       24 | Rock Concert       |
|       25 | Fashion Show       |
|       26 | Sports Event       |
|       27 | Concert Night      |
|       28 | Movie Night        |
|       29 | Tech Conference    |
|       30 | Cultural Festival  |
+----------+--------------------+
10 rows in set (0.00 sec)
```

3. Write a SQL query to select events with available tickets.



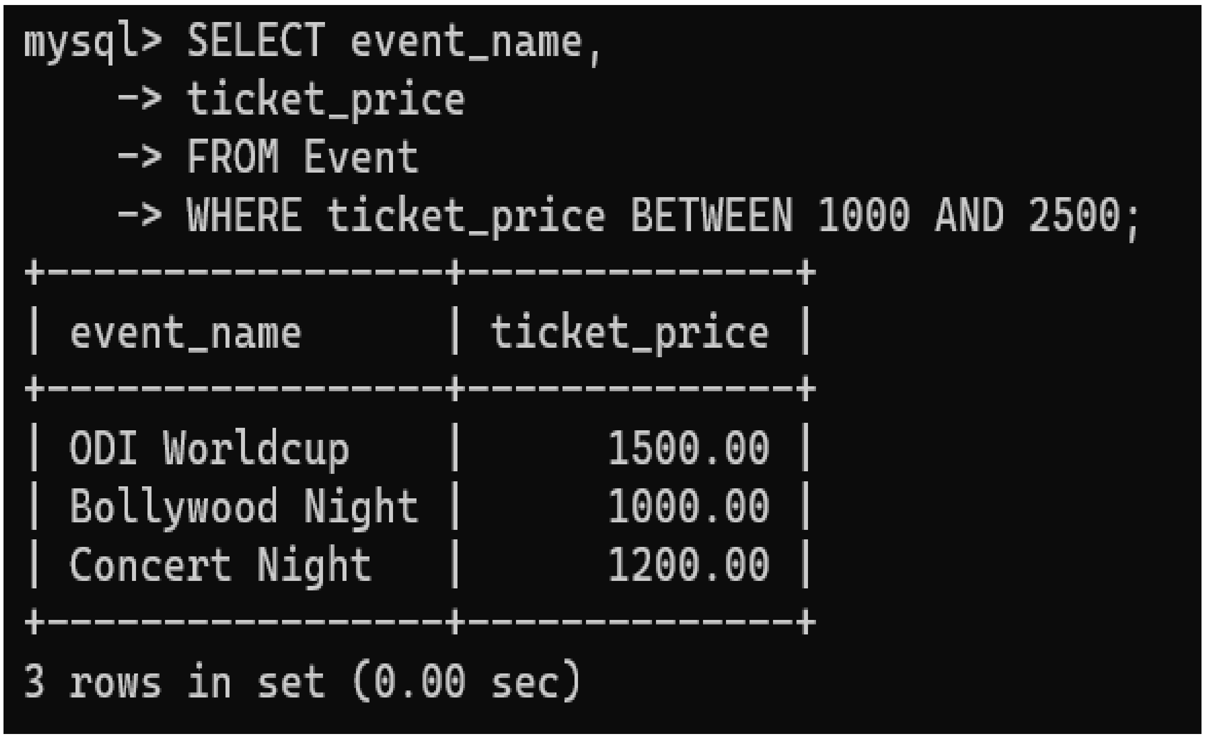
```
mysql> SELECT event_id,event_name
    -> FROM Event
    -> WHERE available_seats > 0;
+----------+-------------------+
| event_id | event_name        |
+----------+-------------------+
|       21 | Cricket Match     |
|       22 | Bollywood Night   |
|       23 | Movie Premiere    |
|       24 | Rock Concert      |
|       25 | Fashion Show      |
|       26 | Sports Event      |
|       27 | Concert Night     |
|       28 | Movie Night       |
|       29 | Tech Conference   |
|       30 | Cultural Festival |
+----------+-------------------+
10 rows in set (0.00 sec)
```

4. Write a SQL query to select events name partial match with ‘cup’.



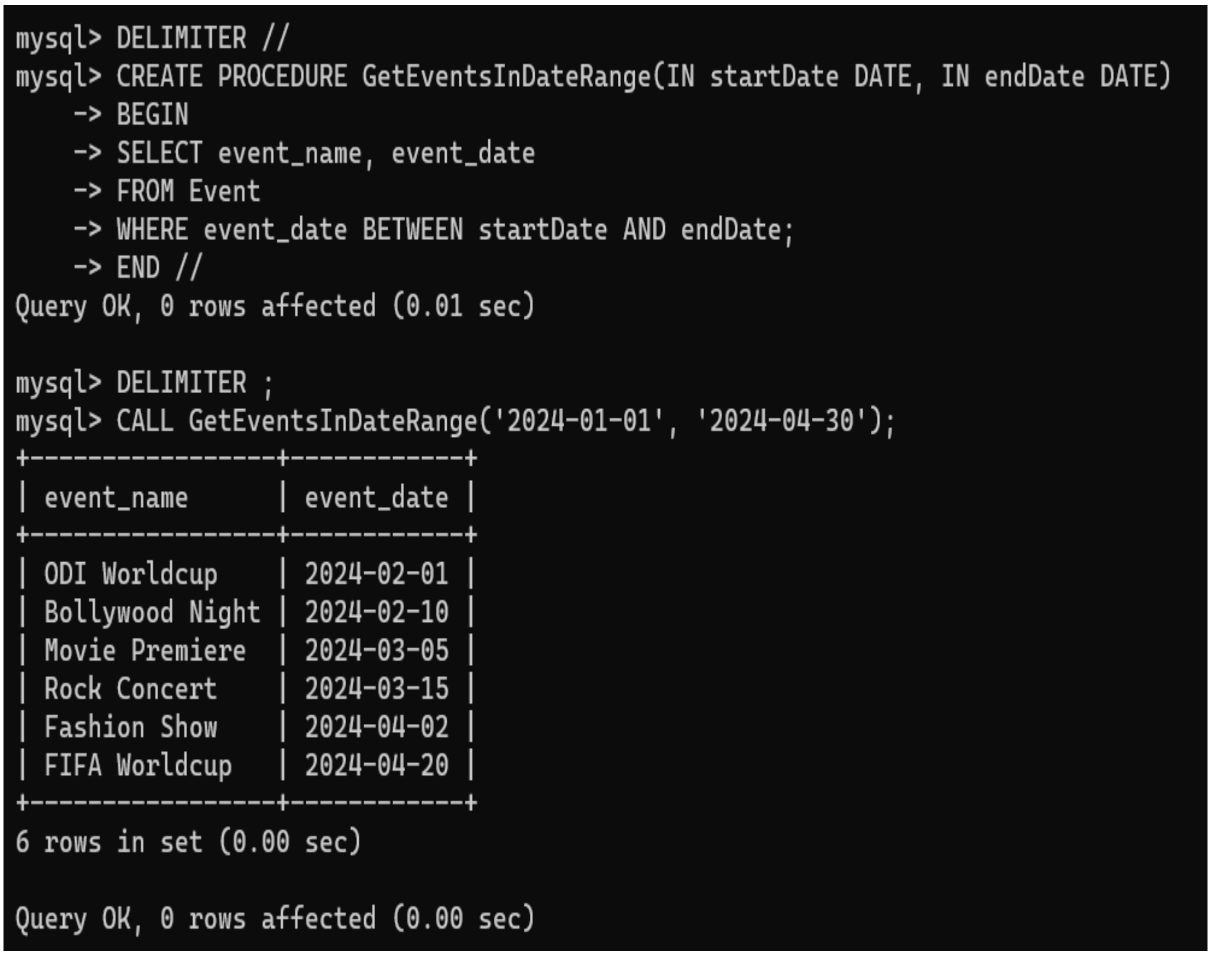
```
mysql> SELECT event_name
    -> FROM Event
    -> WHERE event_name LIKE '%cup%';
+---------------+
| event_name    |
+---------------+
| ODI Worldcup  |
| FIFA Worldcup |
+---------------+
2 rows in set (0.01 sec)
```

5. Write a SQL query to select events with ticket price range is between 1000 to 2500.



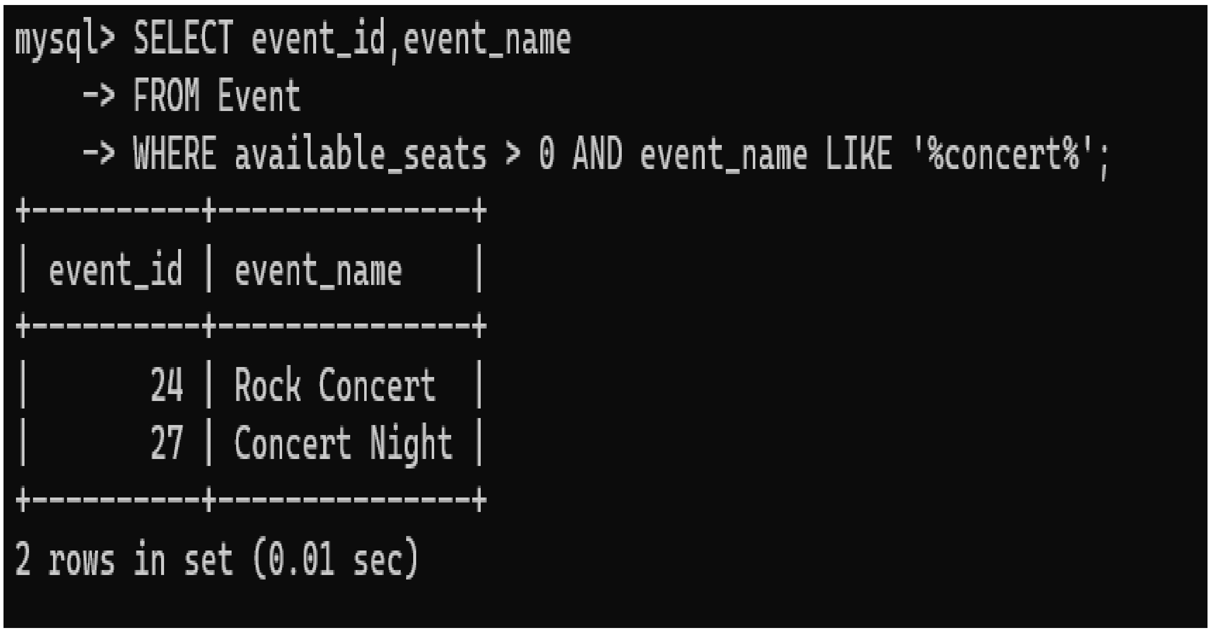
```
mysql> SELECT event_name,
    -> ticket_price
    -> FROM Event
    -> WHERE ticket_price BETWEEN 1000 AND 2500;
+-----------------+--------------+
| event_name      | ticket_price |
+-----------------+--------------+
| ODI Worldcup    |      1500.00 |
| Bollywood Night |      1000.00 |
| Concert Night   |      1200.00 |
+-----------------+--------------+
3 rows in set (0.00 sec)
```

6. Write a SQL query to retrieve events with dates falling within a specific range.



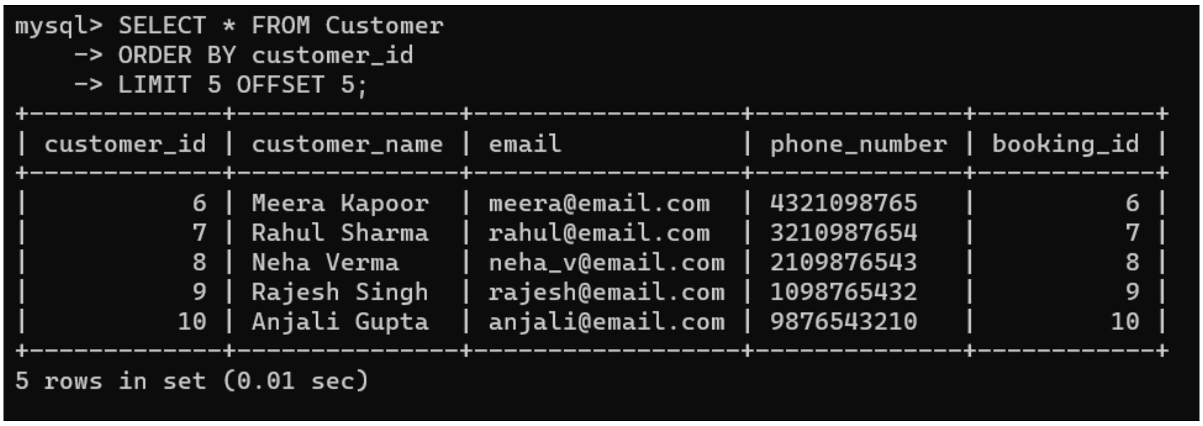
```
mysql> DELIMITER //
mysql> CREATE PROCEDURE GetEventsInDateRange(IN startDate DATE, IN endDate DATE)
    -> BEGIN
    -> SELECT event_name, event_date
    -> FROM Event
    -> WHERE event_date BETWEEN startDate AND endDate;
    -> END //
Query OK, 0 rows affected (0.01 sec)

mysql> DELIMITER ;
mysql> CALL GetEventsInDateRange('2024-01-01', '2024-04-30');
+-----------------+------------+
| event_name      | event_date |
+-----------------+------------+
| ODI Worldcup    | 2024-02-01 |
| Bollywood Night | 2024-02-10 |
| Movie Premiere  | 2024-03-05 |
| Rock Concert    | 2024-03-15 |
| Fashion Show    | 2024-04-02 |
| FIFA Worldcup   | 2024-04-20 |
+-----------------+------------+
6 rows in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)
```

7. Write a SQL query to retrieve events with available tickets that also have "Concert" in their name.



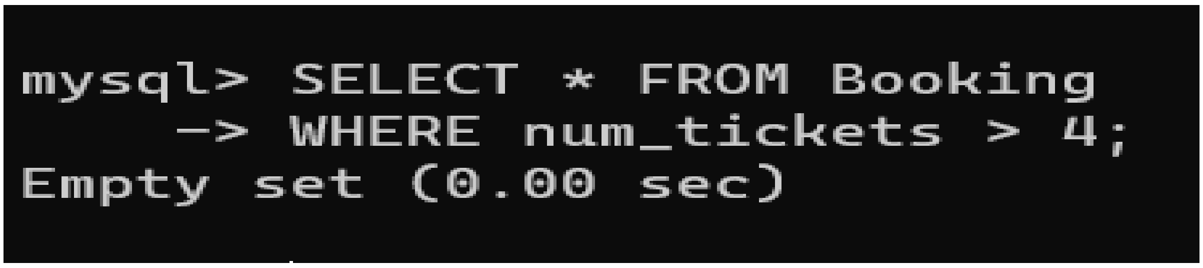
```
mysql> SELECT event_id,event_name
    -> FROM Event
    -> WHERE available_seats > 0 AND event_name LIKE '%concert%';
+----------+---------------+
| event_id | event_name    |
+----------+---------------+
|       24 | Rock Concert  |
|       27 | Concert Night |
+----------+---------------+
2 rows in set (0.01 sec)
```

8. Write a SQL query to retrieve users in batches of 5, starting from the 6th user.



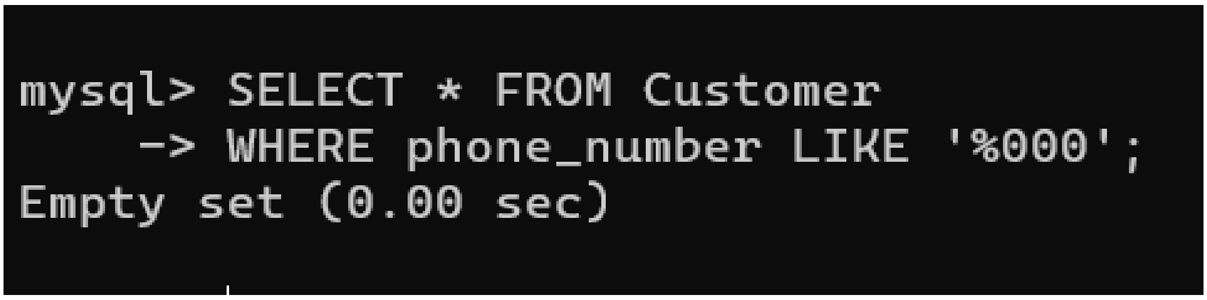
```
mysql> SELECT * FROM Customer
    -> ORDER BY customer_id
    -> LIMIT 5 OFFSET 5;
+-------------+---------------+------------------+--------------+------------+
| customer_id | customer_name | email            | phone_number | booking_id |
+-------------+---------------+------------------+--------------+------------+
|           6 | Meera Kapoor  | meera@email.com  | 4321098765   |          6 |
|           7 | Rahul Sharma  | rahul@email.com  | 3210987654   |          7 |
|           8 | Neha Verma    | neha_v@email.com | 2109876543   |          8 |
|           9 | Rajesh Singh  | rajesh@email.com | 1098765432   |          9 |
|          10 | Anjali Gupta  | anjali@email.com | 9876543210   |         10 |
+-------------+---------------+------------------+--------------+------------+
5 rows in set (0.01 sec)
```

9. Write a SQL query to retrieve bookings details contains booked no of ticket more than 4.



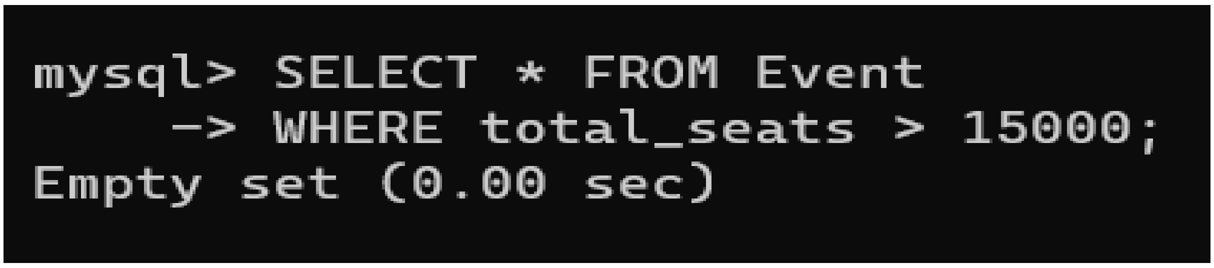
```
mysql> SELECT * FROM Booking
    -> WHERE num_tickets > 4;
Empty set (0.00 sec)
```

10. Write a SQL query to retrieve customer information whose phone number end with ‘000’



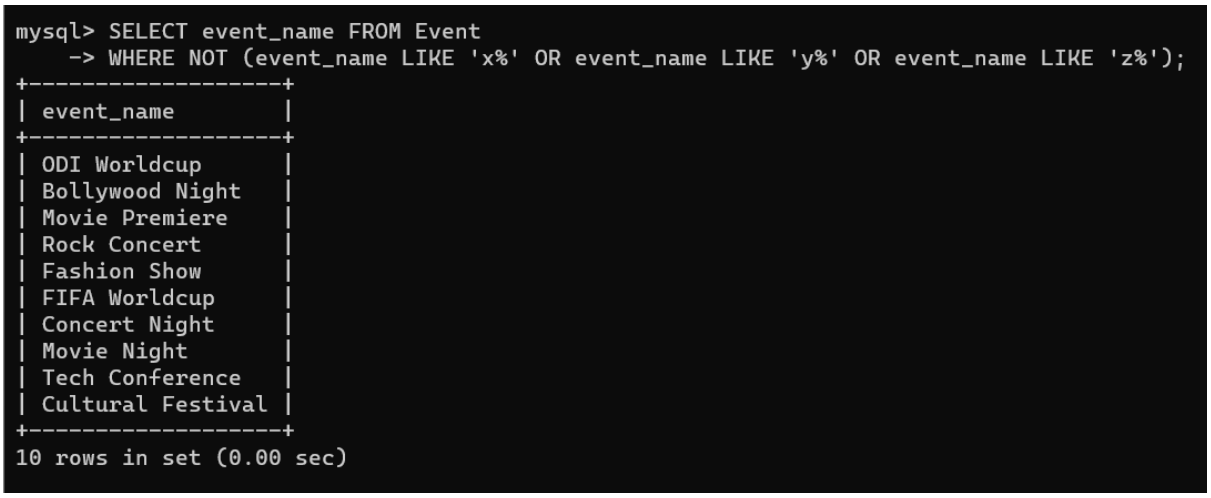
```
mysql> SELECT * FROM Customer
    -> WHERE phone_number LIKE '%000';
Empty set (0.00 sec)
```

11. Write a SQL query to retrieve the events in order whose seat capacity more than 15000.



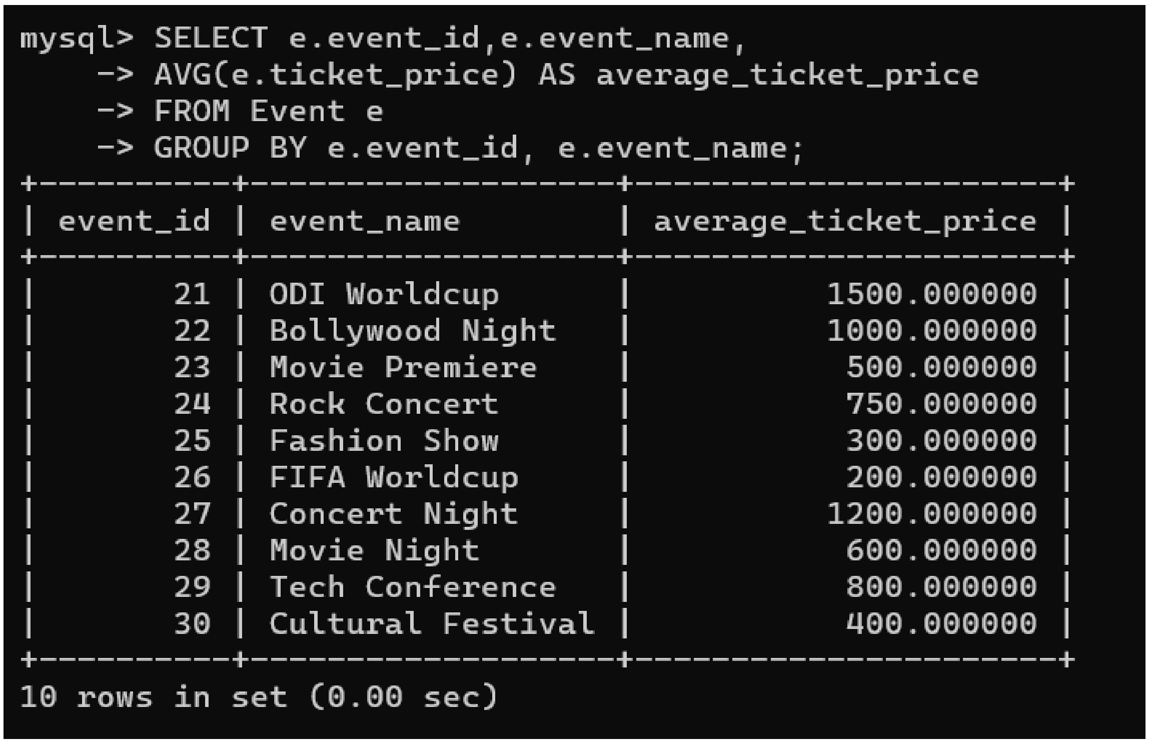
```
mysql> SELECT * FROM Event
    -> WHERE total_seats > 15000;
Empty set (0.00 sec)
```

12. Write a SQL query to select events name not start with ‘x’, ‘y’, ‘z’


```
mysql> SELECT event_name FROM Event
    -> WHERE NOT (event_name LIKE 'x%' OR event_name LIKE 'y%' OR event_name LIKE 'z%');
+--------------------+
| event_name         |
+--------------------+
| ODI Worldcup       |
| Bollywood Night    |
| Movie Premiere     |
| Rock Concert       |
| Fashion Show       |
| FIFA Worldcup      |
| Concert Night      |
| Movie Night        |
| Tech Conference    |
| Cultural Festival  |
+--------------------+
10 rows in set (0.00 sec)
```

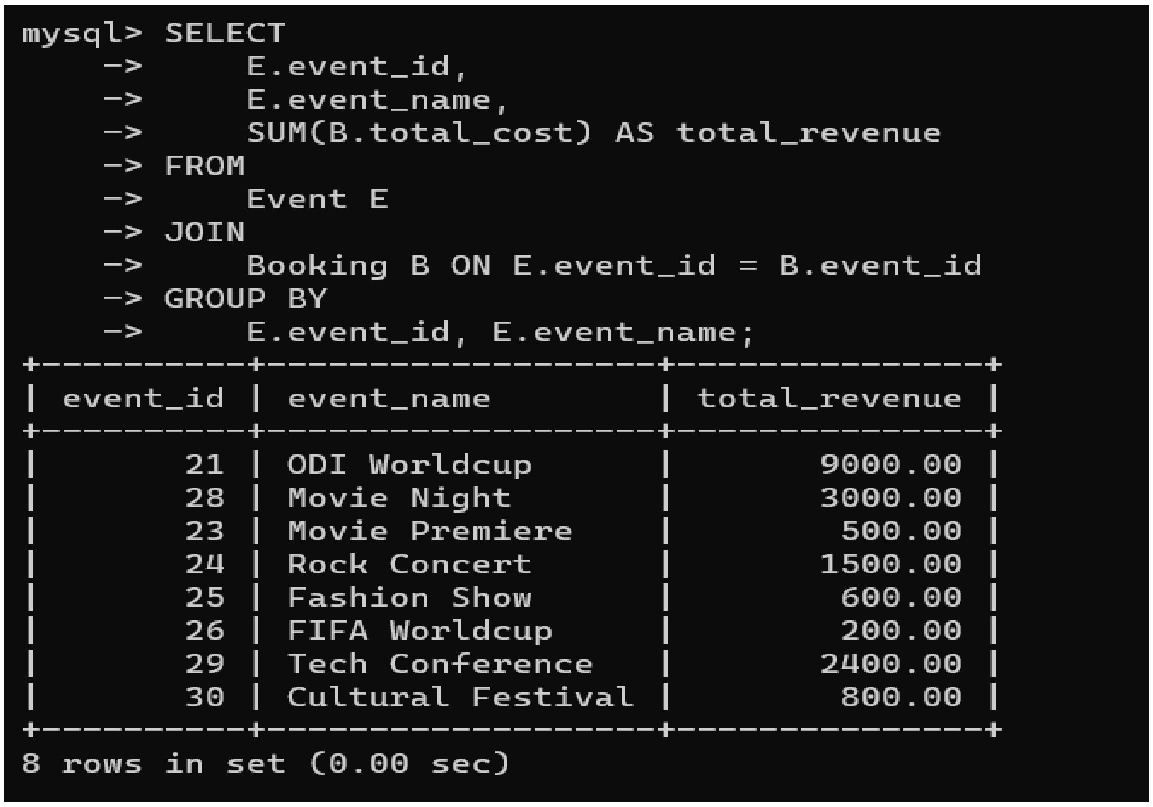
Tasks 3: Aggregate functions, Having, Order By, GroupBy and Joins

1. Write a SQL query to List Events and Their Average Ticket Prices.



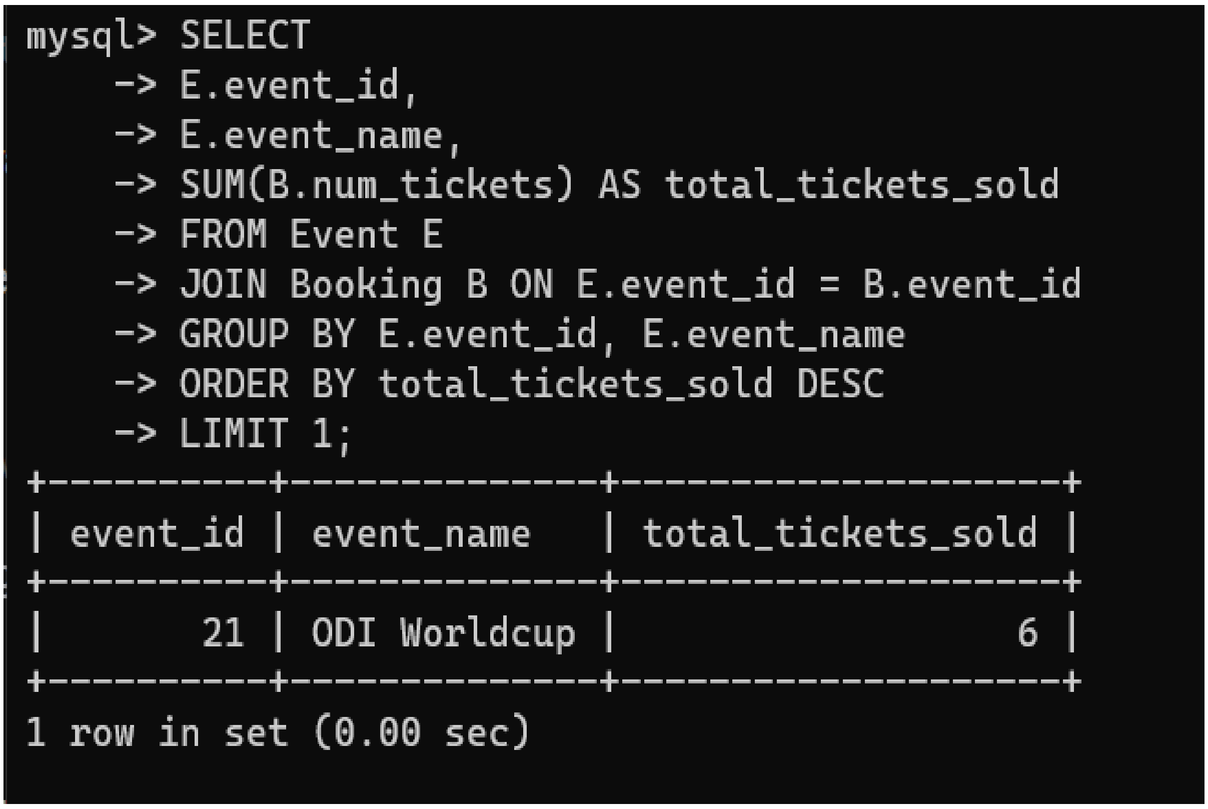
```
mysql> SELECT e.event_id,e.event_name,
    -> AVG(e.ticket_price) AS average_ticket_price
    -> FROM Event e
    -> GROUP BY e.event_id, e.event_name;
+----------+-------------------+----------------------+
| event_id | event_name        | average_ticket_price |
+----------+-------------------+----------------------+
|       21 | ODI Worldcup      |          1500.000000 |
|       22 | Bollywood Night   |          1000.000000 |
|       23 | Movie Premiere    |           500.000000 |
|       24 | Rock Concert      |           750.000000 |
|       25 | Fashion Show      |           300.000000 |
|       26 | FIFA Worldcup     |           200.000000 |
|       27 | Concert Night     |          1200.000000 |
|       28 | Movie Night       |           600.000000 |
|       29 | Tech Conference   |           800.000000 |
|       30 | Cultural Festival |           400.000000 |
+----------+-------------------+----------------------+
10 rows in set (0.00 sec)
```

2. Write a SQL query to Calculate the Total Revenue Generated by Events.



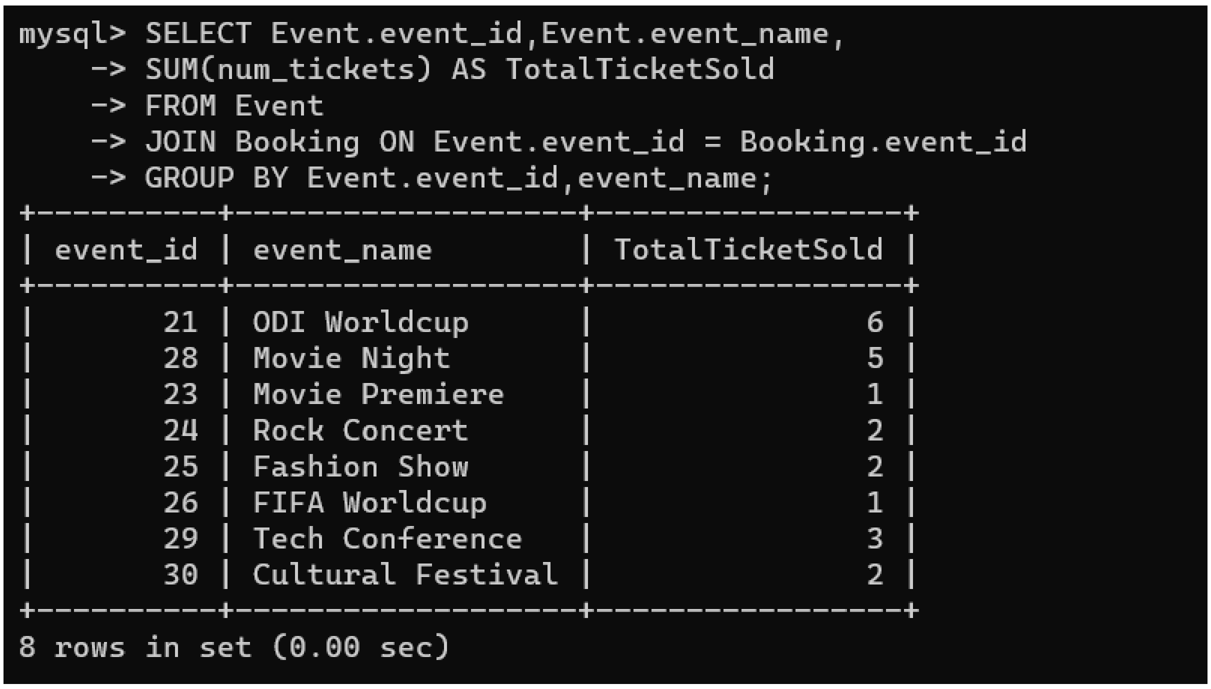
```
mysql> SELECT
    ->     E.event_id,
    ->     E.event_name,
    ->     SUM(B.total_cost) AS total_revenue
    -> FROM
    ->     Event E
    -> JOIN
    ->     Booking B ON E.event_id = B.event_id
    -> GROUP BY
    ->     E.event_id, E.event_name;
+----------+-------------------+---------------+
| event_id | event_name        | total_revenue |
+----------+-------------------+---------------+
|       21 | ODI Worldcup      |       9000.00 |
|       28 | Movie Night       |       3000.00 |
|       23 | Movie Premiere    |        500.00 |
|       24 | Rock Concert      |       1500.00 |
|       25 | Fashion Show      |        600.00 |
|       26 | FIFA Worldcup     |        200.00 |
|       29 | Tech Conference   |       2400.00 |
|       30 | Cultural Festival |        800.00 |
+----------+-------------------+---------------+
8 rows in set (0.00 sec)
```

3. Write a SQL query to find the event with the highest ticket sales.



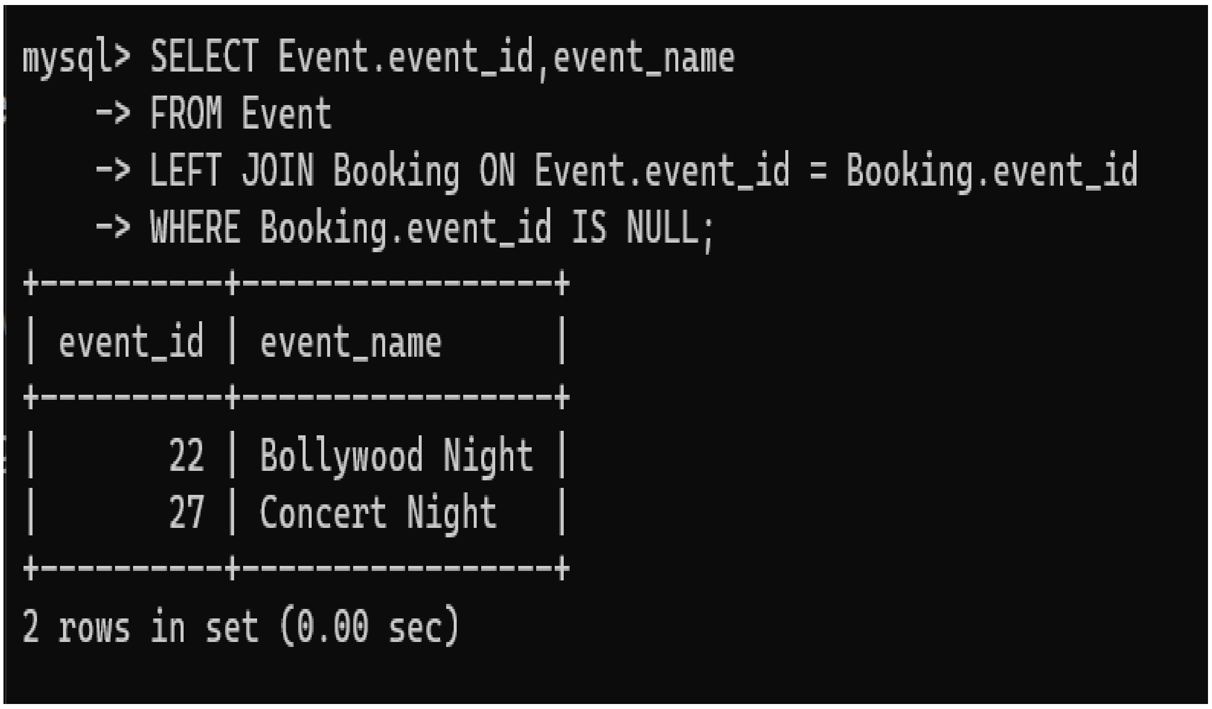
```
mysql> SELECT
    -> E.event_id,
    -> E.event_name,
    -> SUM(B.num_tickets) AS total_tickets_sold
    -> FROM Event E
    -> JOIN Booking B ON E.event_id = B.event_id
    -> GROUP BY E.event_id, E.event_name
    -> ORDER BY total_tickets_sold DESC
    -> LIMIT 1;
+----------+--------------+--------------------+
| event_id | event_name   | total_tickets_sold |
+----------+--------------+--------------------+
|       21 | ODI Worldcup |                  6 |
+----------+--------------+--------------------+
1 row in set (0.00 sec)
```

4. Write a SQL query to Calculate the Total Number of Tickets Sold for Each Event.



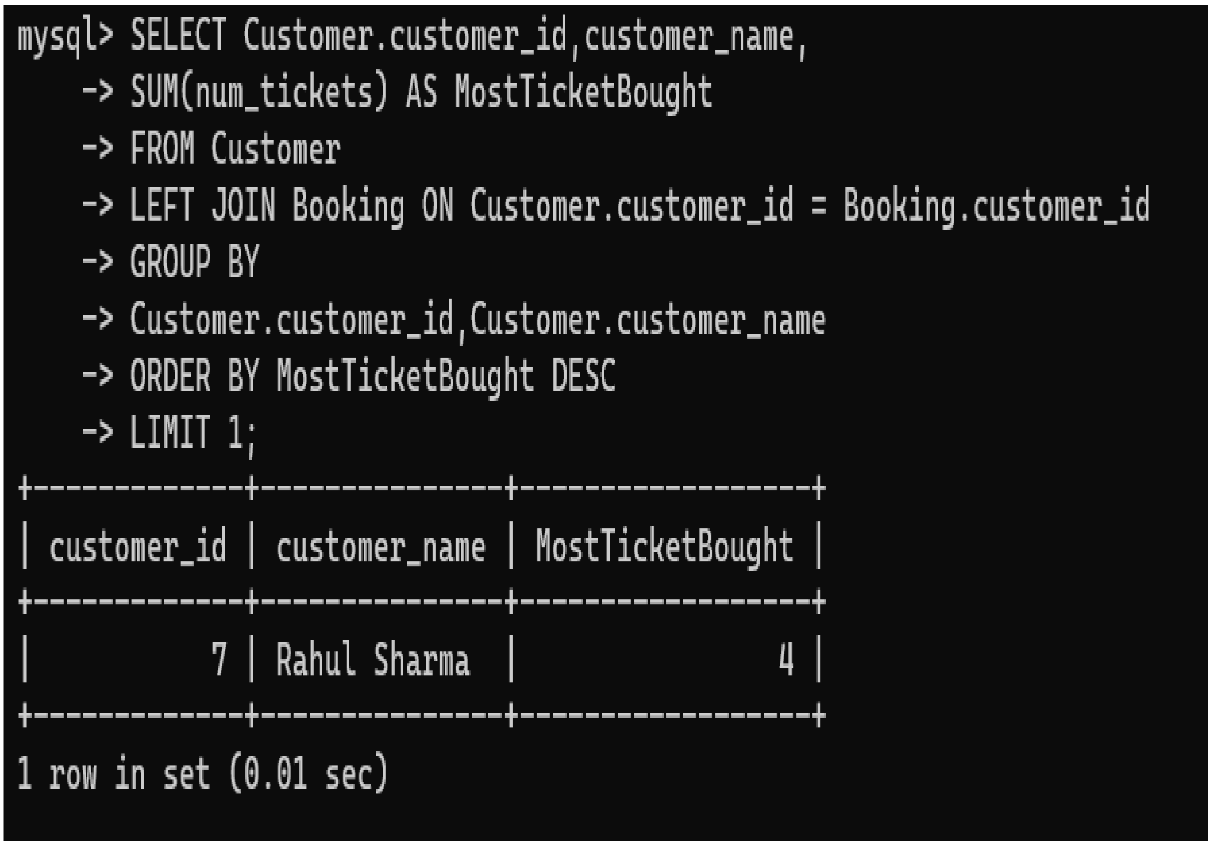
```
mysql> SELECT Event.event_id,Event.event_name,
    -> SUM(num_tickets) AS TotalTicketSold
    -> FROM Event
    -> JOIN Booking ON Event.event_id = Booking.event_id
    -> GROUP BY Event.event_id,event_name;
+----------+-------------------+-----------------+
| event_id | event_name        | TotalTicketSold |
+----------+-------------------+-----------------+
|       21 | ODI Worldcup      |               6 |
|       28 | Movie Night       |               5 |
|       23 | Movie Premiere    |               1 |
|       24 | Rock Concert      |               2 |
|       25 | Fashion Show      |               2 |
|       26 | FIFA Worldcup     |               1 |
|       29 | Tech Conference   |               3 |
|       30 | Cultural Festival |               2 |
+----------+-------------------+-----------------+
8 rows in set (0.00 sec)
```

5. Write a SQL query to Find Events with No Ticket Sales.



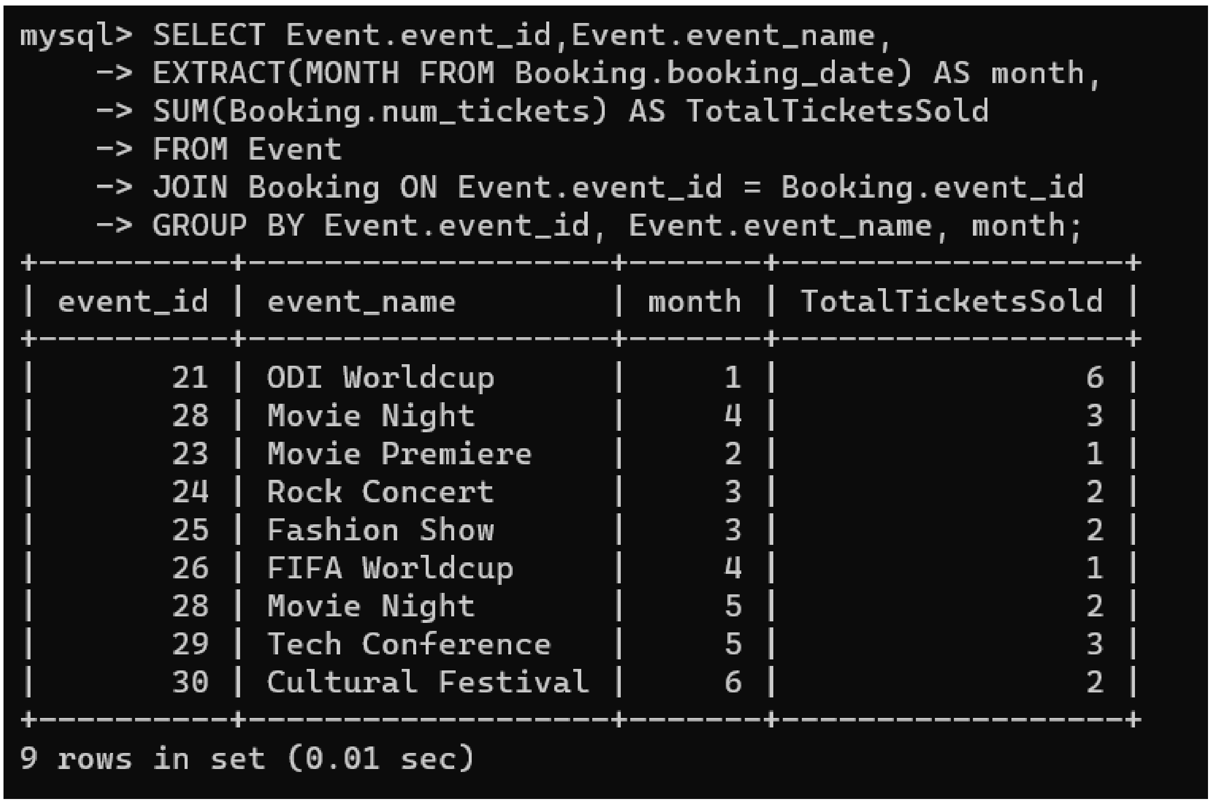
```
mysql> SELECT Event.event_id,event_name
    -> FROM Event
    -> LEFT JOIN Booking ON Event.event_id = Booking.event_id
    -> WHERE Booking.event_id IS NULL;
+----------+-----------------+
| event_id | event_name      |
+----------+-----------------+
|       22 | Bollywood Night |
|       27 | Concert Night   |
+----------+-----------------+
2 rows in set (0.00 sec)
```

6. Write a SQL query to Find the User Who Has Booked the Most Tickets.



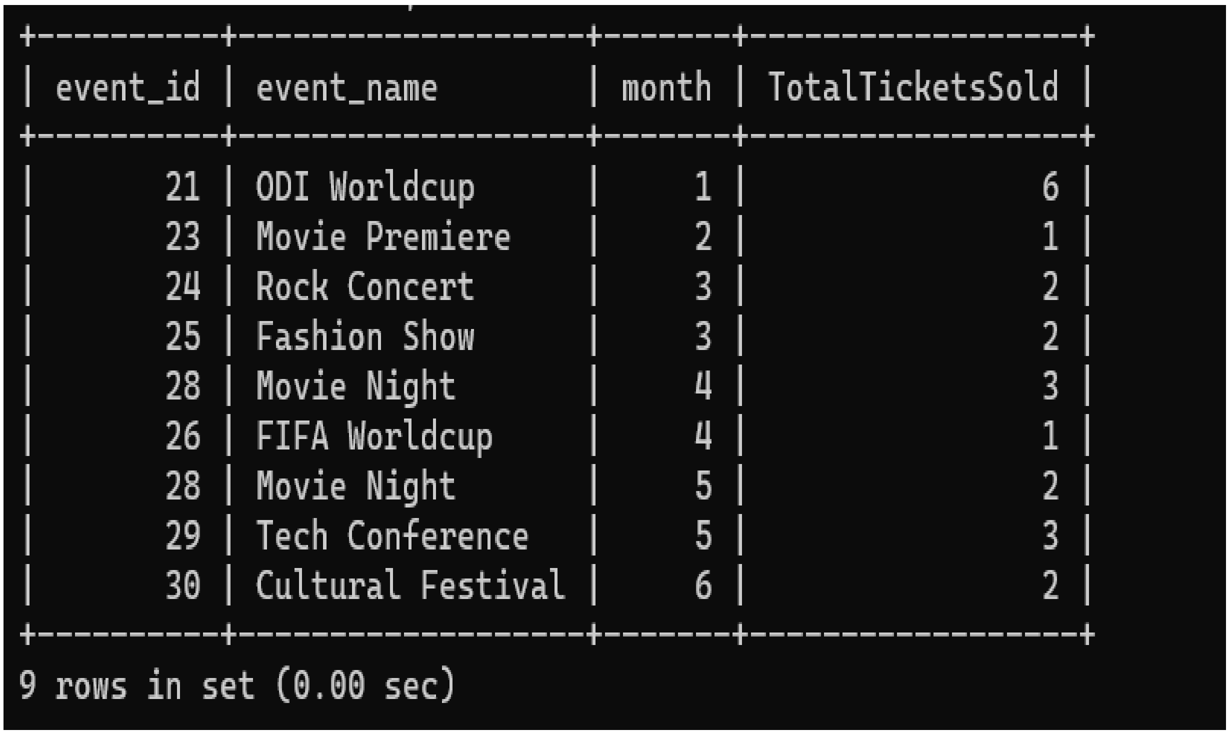
```
mysql> SELECT Customer.customer_id,customer_name,
    -> SUM(num_tickets) AS MostTicketBought
    -> FROM Customer
    -> LEFT JOIN Booking ON Customer.customer_id = Booking.customer_id
    -> GROUP BY
    -> Customer.customer_id,Customer.customer_name
    -> ORDER BY MostTicketBought DESC
    -> LIMIT 1;
+-------------+---------------+------------------+
| customer_id | customer_name | MostTicketBought |
+-------------+---------------+------------------+
|           7 | Rahul Sharma  |                4 |
+-------------+---------------+------------------+
1 row in set (0.01 sec)
```

7. Write a SQL query to List Events and the total number of tickets sold for each month.



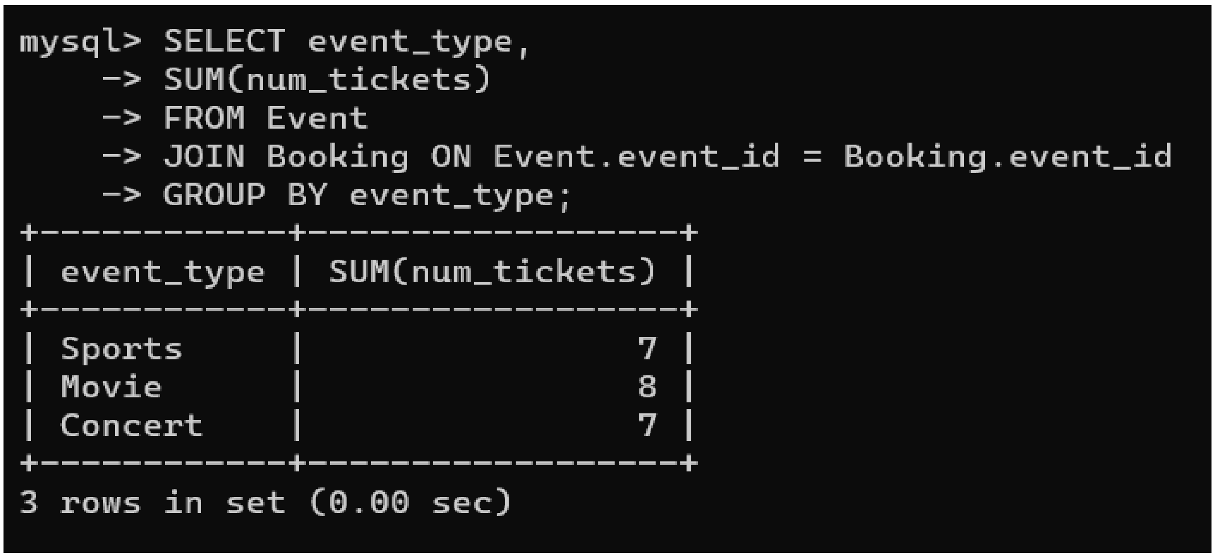
```
mysql> SELECT Event.event_id,Event.event_name,
    -> EXTRACT(MONTH FROM Booking.booking_date) AS month,
    -> SUM(Booking.num_tickets) AS TotalTicketsSold
    -> FROM Event
    -> JOIN Booking ON Event.event_id = Booking.event_id
    -> GROUP BY Event.event_id, Event.event_name, month;
+----------+-------------------+-------+------------------+
| event_id | event_name        | month | TotalTicketsSold |
+----------+-------------------+-------+------------------+
|       21 | ODI Worldcup      |     1 |                6 |
|       28 | Movie Night       |     4 |                3 |
|       23 | Movie Premiere    |     2 |                1 |
|       24 | Rock Concert      |     3 |                2 |
|       25 | Fashion Show      |     3 |                2 |
|       26 | FIFA Worldcup     |     4 |                1 |
|       28 | Movie Night       |     5 |                2 |
|       29 | Tech Conference   |     5 |                3 |
|       30 | Cultural Festival |     6 |                2 |
+----------+-------------------+-------+------------------+
9 rows in set (0.01 sec)
```

8. Write a SQL query to calculate the average Ticket Price for Events in Each Venue.



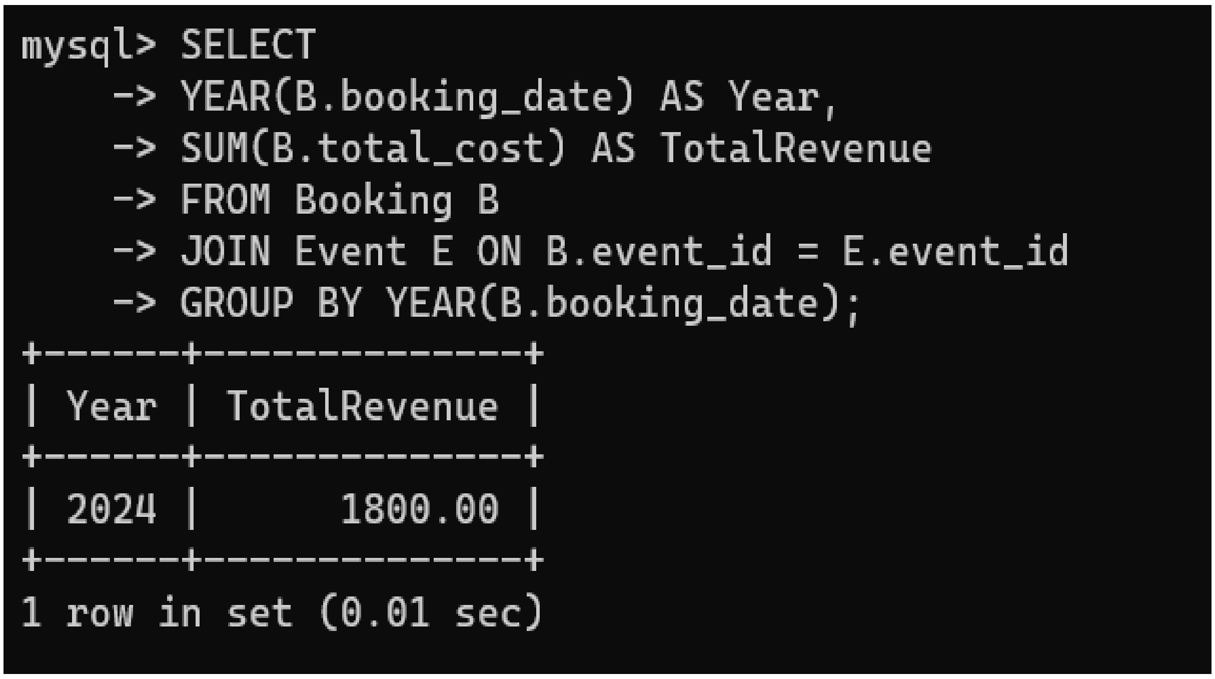
```
+----------+-------------------+-------+------------------+
| event_id | event_name        | month | TotalTicketsSold |
+----------+-------------------+-------+------------------+
|       21 | ODI Worldcup      |     1 |                6 |
|       23 | Movie Premiere    |     2 |                1 |
|       24 | Rock Concert      |     3 |                2 |
|       25 | Fashion Show      |     3 |                2 |
|       28 | Movie Night       |     4 |                3 |
|       26 | FIFA Worldcup     |     4 |                1 |
|       28 | Movie Night       |     5 |                2 |
|       29 | Tech Conference   |     5 |                3 |
|       30 | Cultural Festival |     6 |                2 |
+----------+-------------------+-------+------------------+
9 rows in set (0.00 sec)
```

9. Write a SQL query to calculate the total Number of Tickets Sold for Each Event Type.



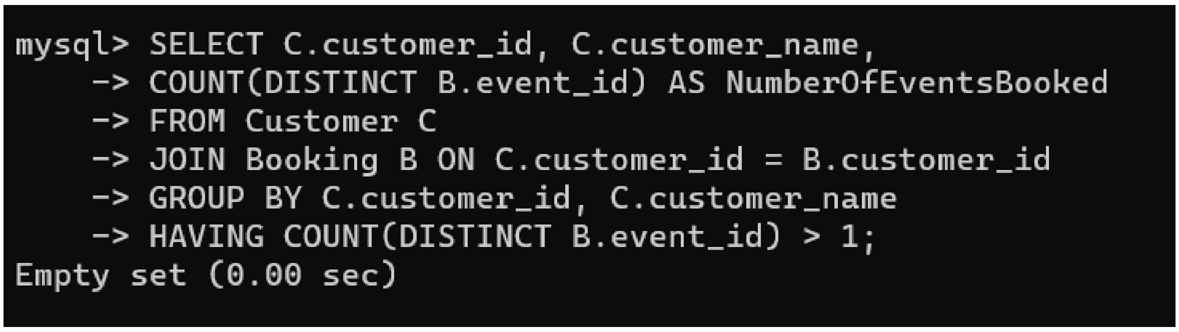
```
mysql> SELECT event_type,
    -> SUM(num_tickets)
    -> FROM Event
    -> JOIN Booking ON Event.event_id = Booking.event_id
    -> GROUP BY event_type;
+------------+------------------+
| event_type | SUM(num_tickets) |
+------------+------------------+
| Sports     |                7 |
| Movie      |                8 |
| Concert    |                7 |
+------------+------------------+
3 rows in set (0.00 sec)
```

10. Write a SQL query to calculate the total Revenue Generated by Events in Each Year.



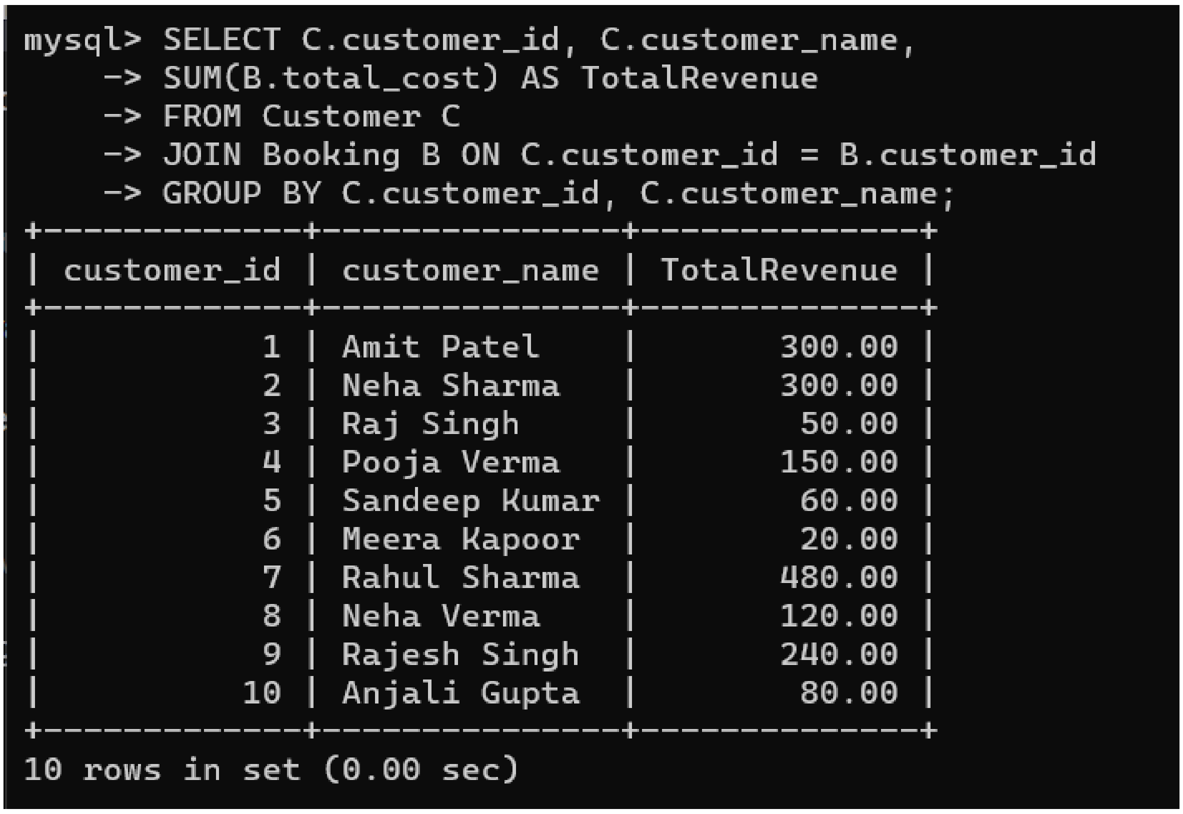
```
mysql> SELECT
    -> YEAR(B.booking_date) AS Year,
    -> SUM(B.total_cost) AS TotalRevenue
    -> FROM Booking B
    -> JOIN Event E ON B.event_id = E.event_id
    -> GROUP BY YEAR(B.booking_date);
+------+--------------+
| Year | TotalRevenue |
+------+--------------+
| 2024 |      1800.00 |
+------+--------------+
1 row in set (0.01 sec)
```

11. Write a SQL query to list users who have booked tickets for multiple events.



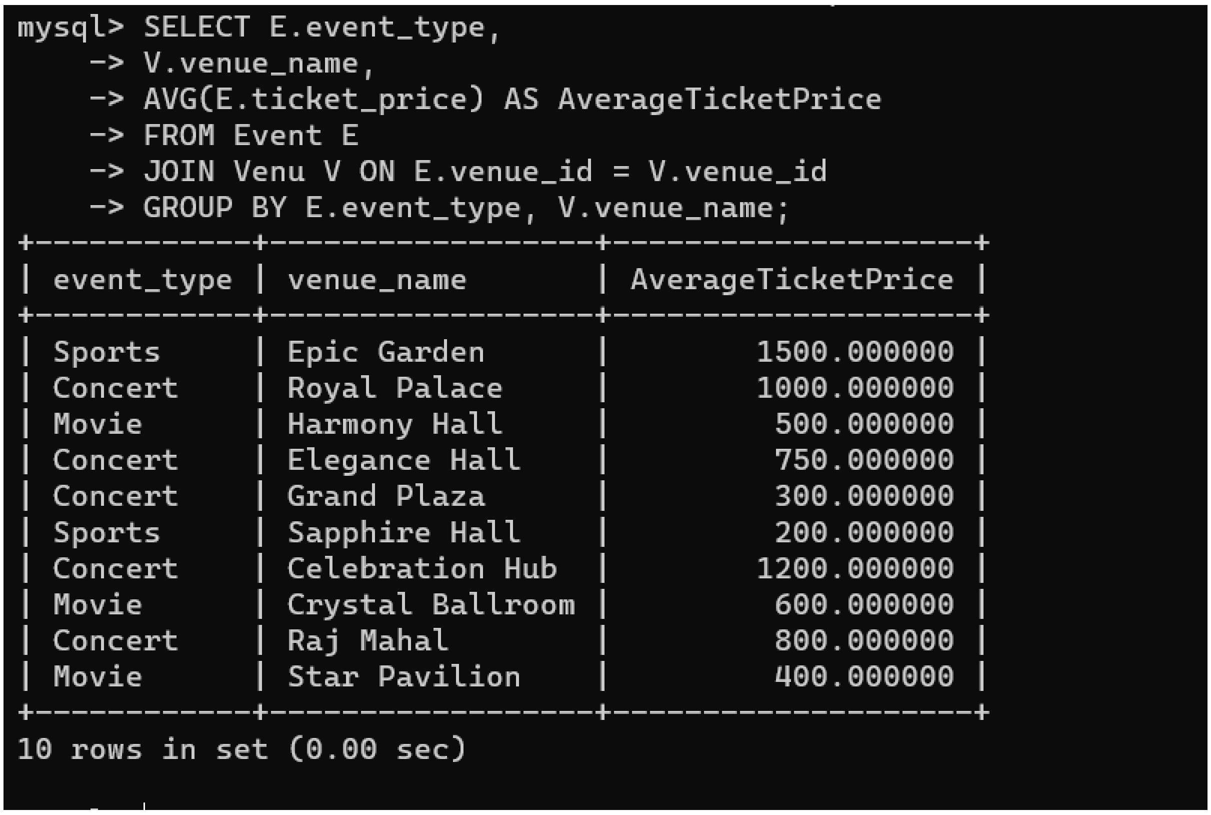
```
mysql> SELECT C.customer_id, C.customer_name,
    -> COUNT(DISTINCT B.event_id) AS NumberOfEventsBooked
    -> FROM Customer C
    -> JOIN Booking B ON C.customer_id = B.customer_id
    -> GROUP BY C.customer_id, C.customer_name
    -> HAVING COUNT(DISTINCT B.event_id) > 1;
Empty set (0.00 sec)
```

12.Write a SQL query to calculate the Total Revenue Generated by Events for Each User.



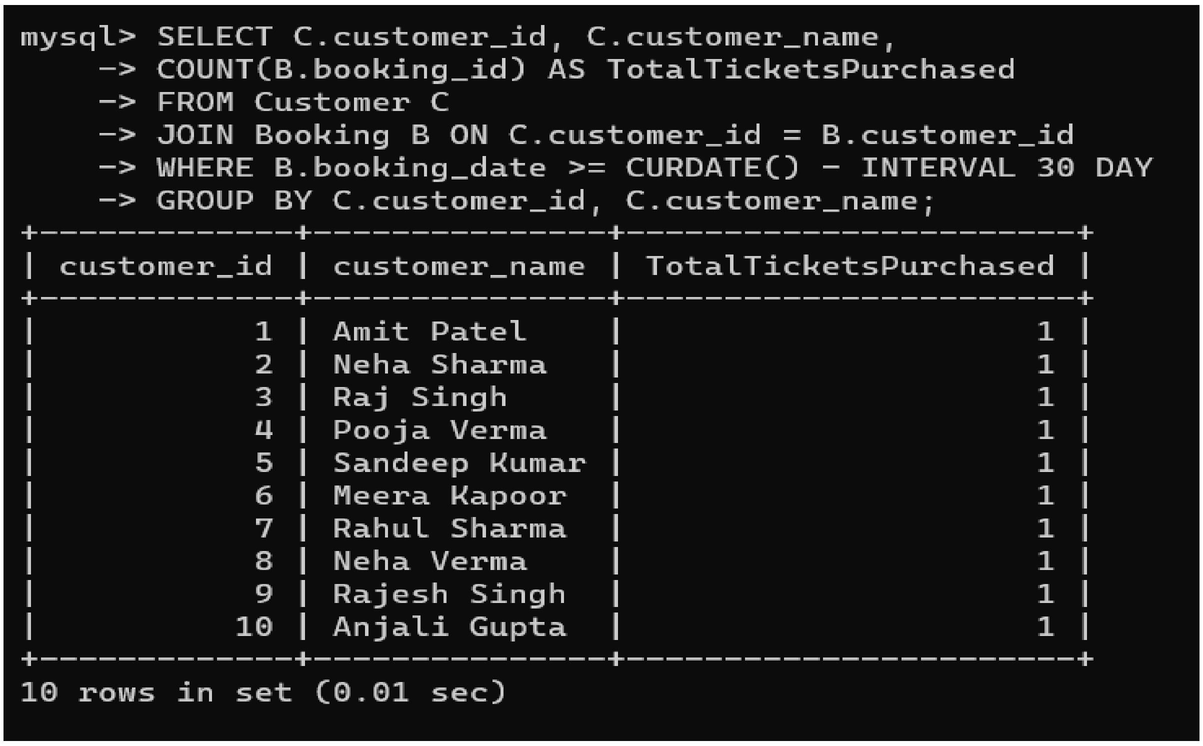
```
mysql> SELECT C.customer_id, C.customer_name,
    -> SUM(B.total_cost) AS TotalRevenue
    -> FROM Customer C
    -> JOIN Booking B ON C.customer_id = B.customer_id
    -> GROUP BY C.customer_id, C.customer_name;
+-------------+---------------+--------------+
| customer_id | customer_name | TotalRevenue |
+-------------+---------------+--------------+
|           1 | Amit Patel    |       300.00 |
|           2 | Neha Sharma   |       300.00 |
|           3 | Raj Singh     |        50.00 |
|           4 | Pooja Verma   |       150.00 |
|           5 | Sandeep Kumar |        60.00 |
|           6 | Meera Kapoor  |        20.00 |
|           7 | Rahul Sharma  |       480.00 |
|           8 | Neha Verma    |       120.00 |
|           9 | Rajesh Singh  |       240.00 |
|          10 | Anjali Gupta  |        80.00 |
+-------------+---------------+--------------+
10 rows in set (0.00 sec)
```

13. Write a SQL query to calculate the Average Ticket Price for Events in Each Category and Venue.



```
mysql> SELECT E.event_type,
    -> V.venue_name,
    -> AVG(E.ticket_price) AS AverageTicketPrice
    -> FROM Event E
    -> JOIN Venu V ON E.venue_id = V.venue_id
    -> GROUP BY E.event_type, V.venue_name;
+------------+------------------+--------------------+
| event_type | venue_name       | AverageTicketPrice |
+------------+------------------+--------------------+
| Sports     | Epic Garden      |        1500.000000 |
| Concert    | Royal Palace     |        1000.000000 |
| Movie      | Harmony Hall     |         500.000000 |
| Concert    | Elegance Hall    |         750.000000 |
| Concert    | Grand Plaza      |         300.000000 |
| Sports     | Sapphire Hall    |         200.000000 |
| Concert    | Celebration Hub  |        1200.000000 |
| Movie      | Crystal Ballroom |         600.000000 |
| Concert    | Raj Mahal        |         800.000000 |
| Movie      | Star Pavilion    |         400.000000 |
+------------+------------------+--------------------+
10 rows in set (0.00 sec)
```

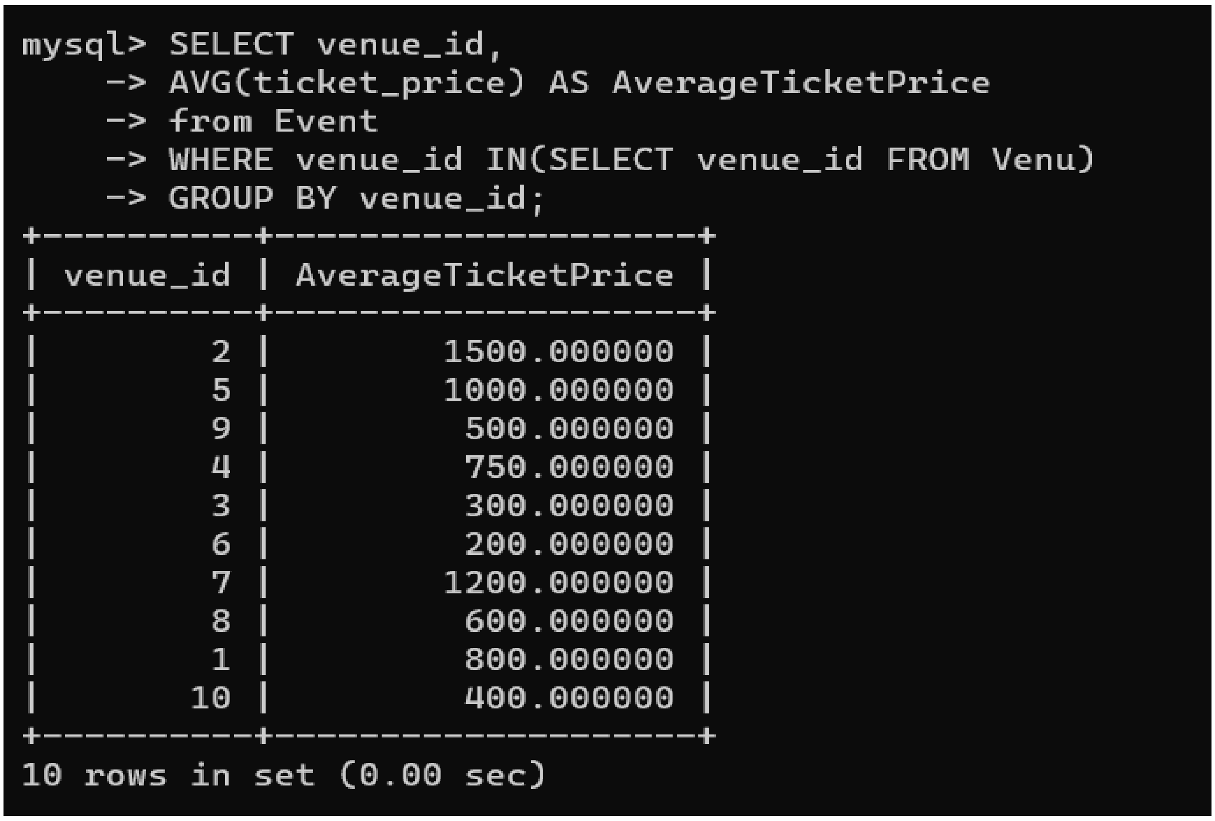
14.Write a SQL query to list Users and the Total Number of Tickets They've Purchased in the Last 30 Days.



```
mysql> SELECT C.customer_id, C.customer_name,
    -> COUNT(B.booking_id) AS TotalTicketsPurchased
    -> FROM Customer C
    -> JOIN Booking B ON C.customer_id = B.customer_id
    -> WHERE B.booking_date >= CURDATE() - INTERVAL 30 DAY
    -> GROUP BY C.customer_id, C.customer_name;
+-------------+---------------+-----------------------+
| customer_id | customer_name | TotalTicketsPurchased |
+-------------+---------------+-----------------------+
|           1 | Amit Patel    |                     1 |
|           2 | Neha Sharma   |                     1 |
|           3 | Raj Singh     |                     1 |
|           4 | Pooja Verma   |                     1 |
|           5 | Sandeep Kumar |                     1 |
|           6 | Meera Kapoor  |                     1 |
|           7 | Rahul Sharma  |                     1 |
|           8 | Neha Verma    |                     1 |
|           9 | Rajesh Singh  |                     1 |
|          10 | Anjali Gupta  |                     1 |
+-------------+---------------+-----------------------+
10 rows in set (0.01 sec)
```

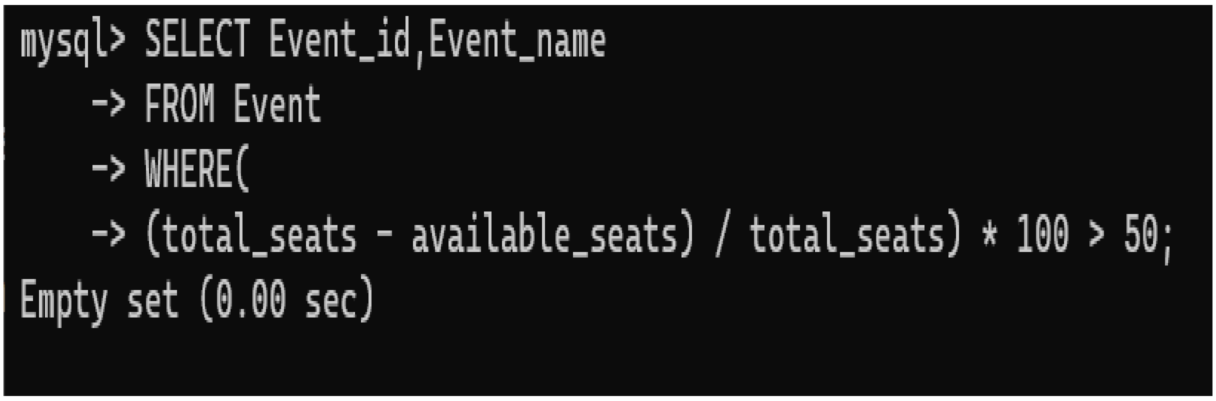
Tasks 4: Subquery and its types

1. Calculate the Average Ticket Price for Events in Each Venue Using a Subquery.



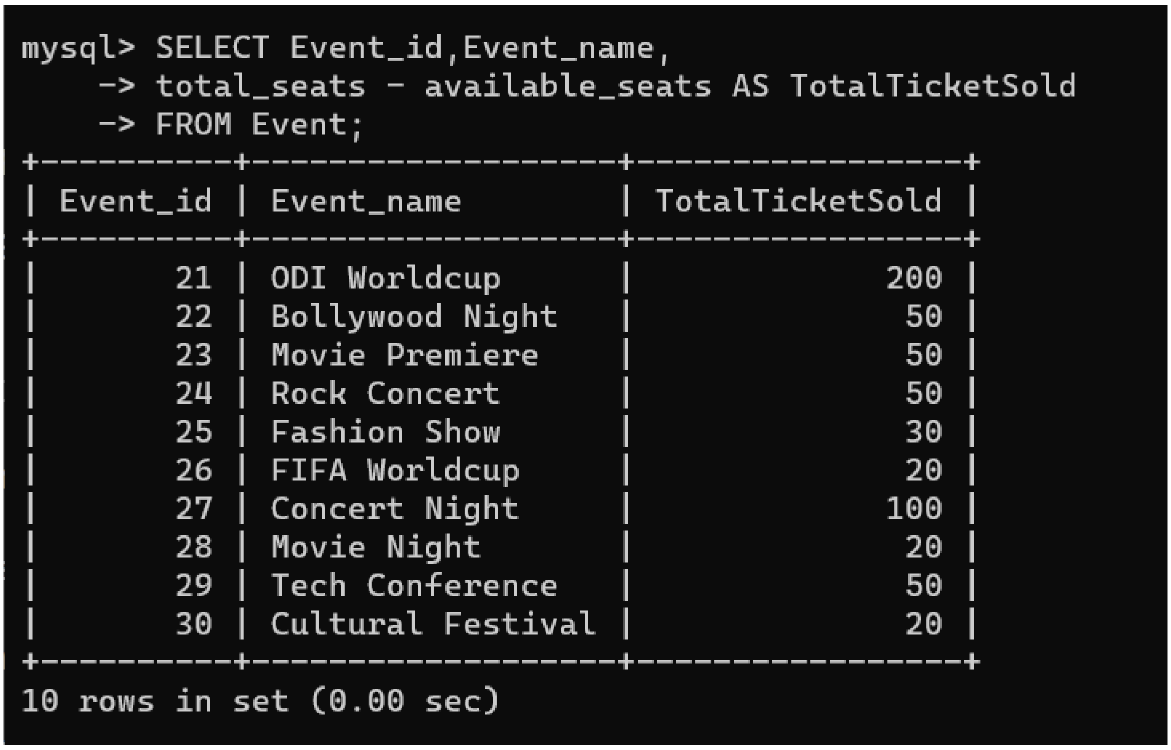
```
mysql> SELECT venue_id,
    -> AVG(ticket_price) AS AverageTicketPrice
    -> from Event
    -> WHERE venue_id IN(SELECT venue_id FROM Venu)
    -> GROUP BY venue_id;
+----------+--------------------+
| venue_id | AverageTicketPrice |
+----------+--------------------+
|        2 |        1500.000000 |
|        5 |        1000.000000 |
|        9 |         500.000000 |
|        4 |         750.000000 |
|        3 |         300.000000 |
|        6 |         200.000000 |
|        7 |        1200.000000 |
|        8 |         600.000000 |
|        1 |         800.000000 |
|       10 |         400.000000 |
+----------+--------------------+
10 rows in set (0.00 sec)
```

2. Find Events with More Than 50% of Tickets Sold using subquery.



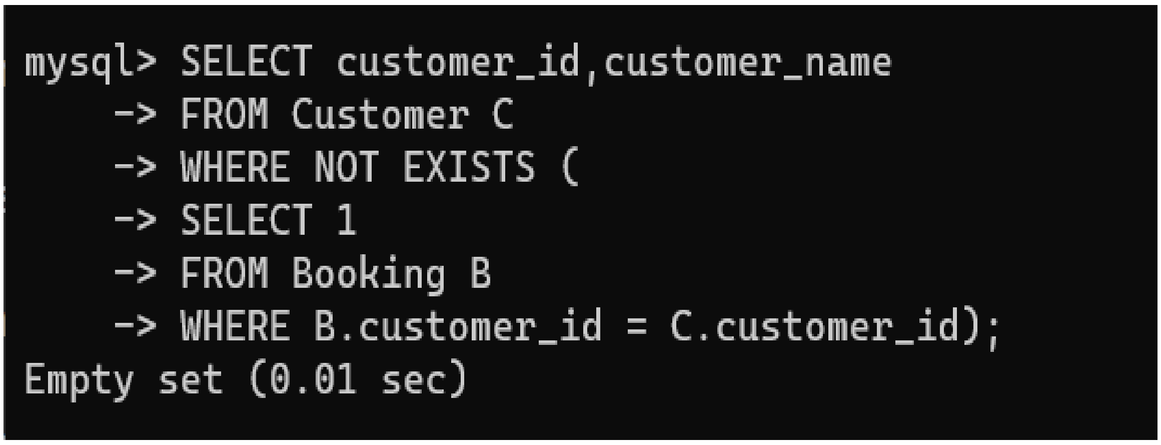
```
mysql> SELECT Event_id,Event_name
    -> FROM Event
    -> WHERE(
    -> (total_seats - available_seats) / total_seats) * 100 > 50;
Empty set (0.00 sec)
```

3. Calculate the Total Number of Tickets Sold for Each Event.



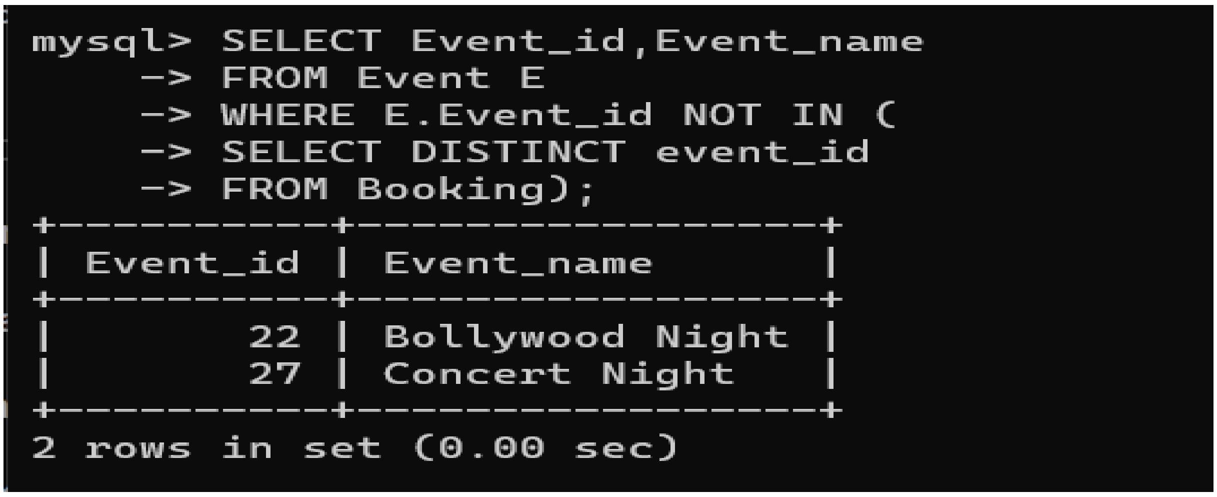
```
mysql> SELECT Event_id,Event_name,
    -> total_seats - available_seats AS TotalTicketSold
    -> FROM Event;
+----------+-------------------+-----------------+
| Event_id | Event_name        | TotalTicketSold |
+----------+-------------------+-----------------+
|       21 | ODI Worldcup      |             200 |
|       22 | Bollywood Night   |              50 |
|       23 | Movie Premiere    |              50 |
|       24 | Rock Concert      |              50 |
|       25 | Fashion Show      |              30 |
|       26 | FIFA Worldcup     |              20 |
|       27 | Concert Night     |             100 |
|       28 | Movie Night       |              20 |
|       29 | Tech Conference   |              50 |
|       30 | Cultural Festival |              20 |
+----------+-------------------+-----------------+
10 rows in set (0.00 sec)
```

4. Find Users Who Have Not Booked Any Tickets Using a NOT EXISTS Subquery.



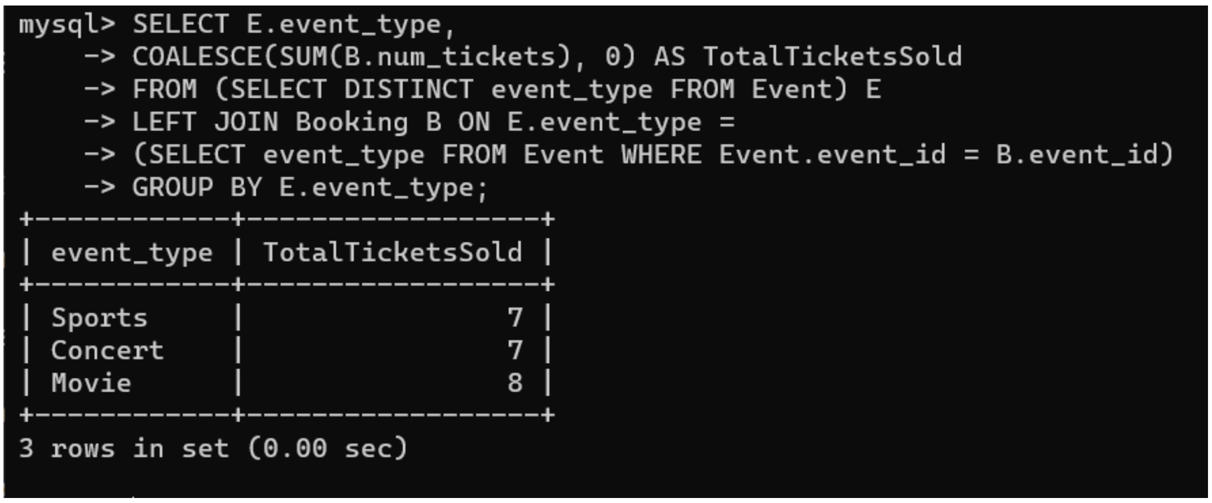
```
mysql> SELECT customer_id,customer_name
    -> FROM Customer C
    -> WHERE NOT EXISTS (
    -> SELECT 1
    -> FROM Booking B
    -> WHERE B.customer_id = C.customer_id);
Empty set (0.01 sec)
```

5. List Events with No Ticket Sales Using a NOT IN Subquery.



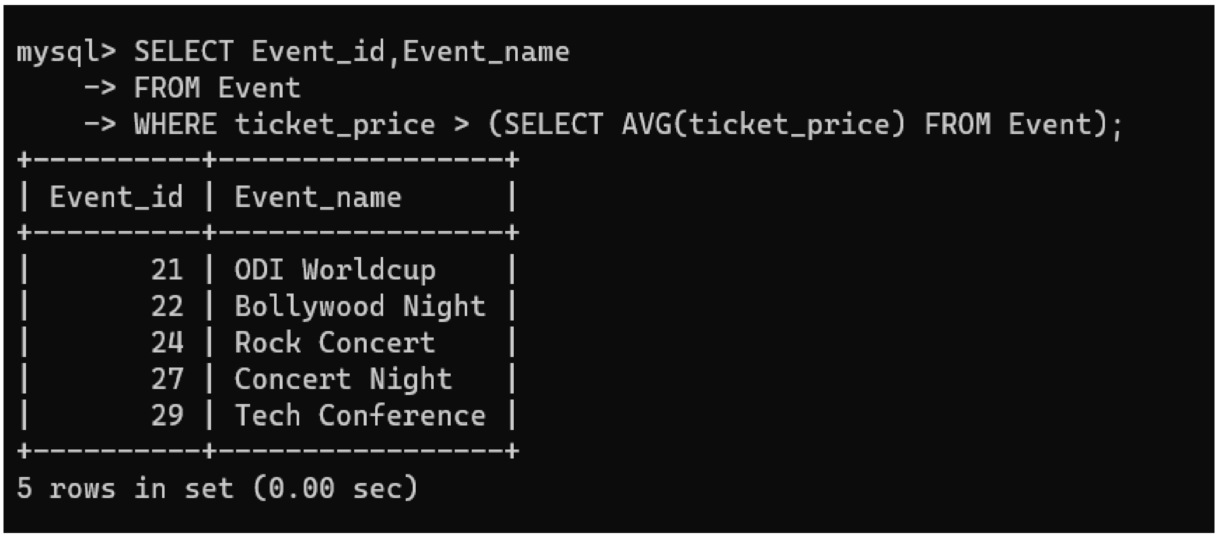
```
mysql> SELECT Event_id,Event_name
    -> FROM Event E
    -> WHERE E.Event_id NOT IN (
    -> SELECT DISTINCT event_id
    -> FROM Booking);
+----------+-----------------+
| Event_id | Event_name      |
+----------+-----------------+
|       22 | Bollywood Night |
|       27 | Concert Night   |
+----------+-----------------+
2 rows in set (0.00 sec)
```

6. Calculate the Total Number of Tickets Sold for Each Event Type Using a Subquery in the FROM Clause



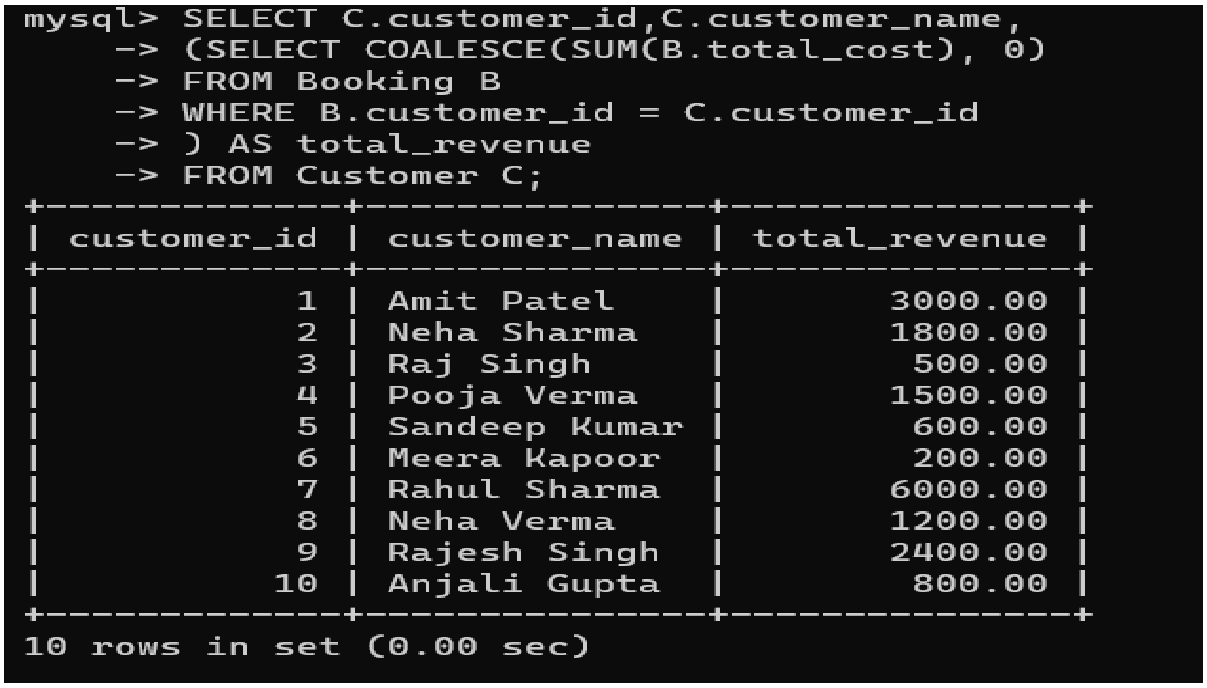
```
mysql> SELECT E.event_type,
    -> COALESCE(SUM(B.num_tickets), 0) AS TotalTicketsSold
    -> FROM (SELECT DISTINCT event_type FROM Event) E
    -> LEFT JOIN Booking B ON E.event_type =
    -> (SELECT event_type FROM Event WHERE Event.event_id = B.event_id)
    -> GROUP BY E.event_type;
+------------+------------------+
| event_type | TotalTicketsSold |
+------------+------------------+
| Sports     |                7 |
| Concert    |                7 |
| Movie      |                8 |
+------------+------------------+
3 rows in set (0.00 sec)
```

7. Find Events with Ticket Prices Higher Than the Average Ticket Price Using a Subquery in the WHERE Clause.



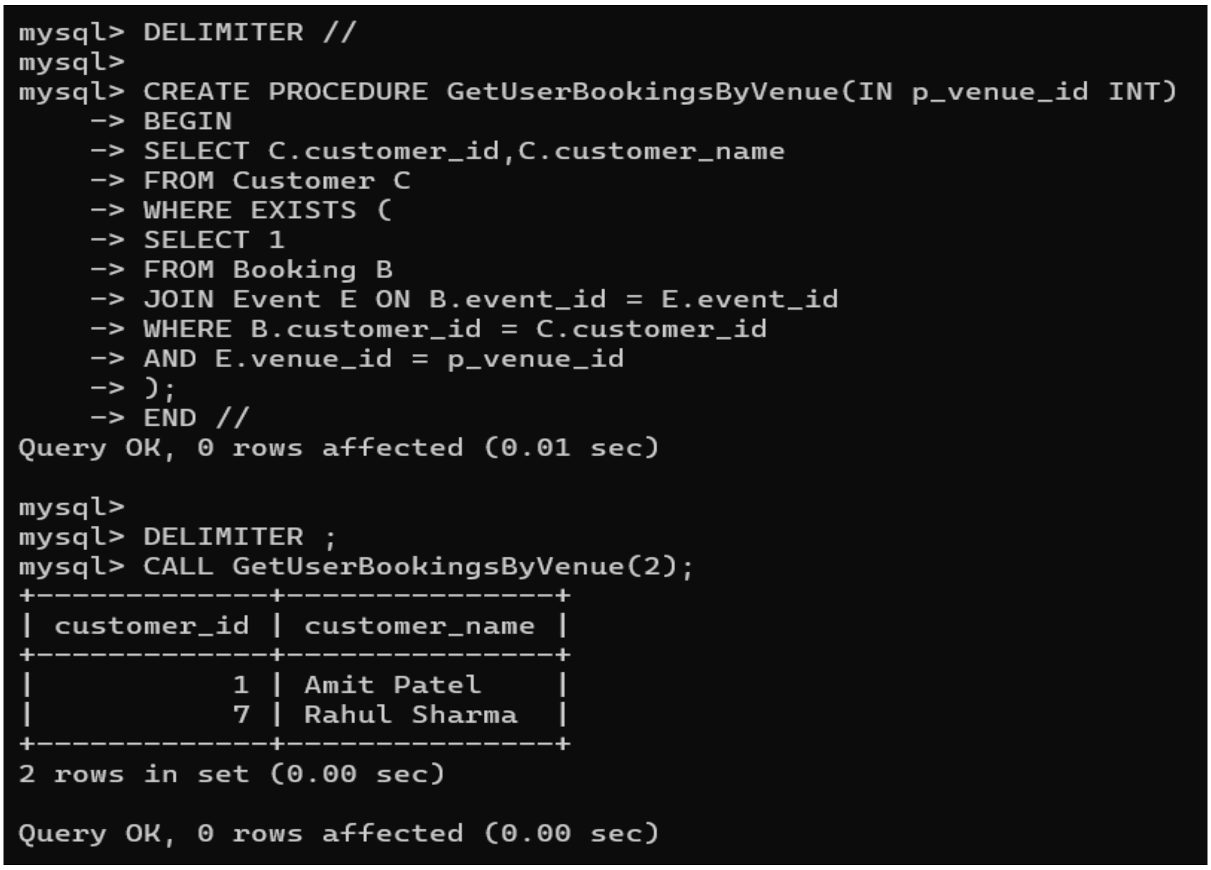
```
mysql> SELECT Event_id,Event_name
    -> FROM Event
    -> WHERE ticket_price > (SELECT AVG(ticket_price) FROM Event);
+----------+-----------------+
| Event_id | Event_name      |
+----------+-----------------+
|       21 | ODI Worldcup    |
|       22 | Bollywood Night |
|       24 | Rock Concert    |
|       27 | Concert Night   |
|       29 | Tech Conference |
+----------+-----------------+
5 rows in set (0.00 sec)
```

8. Calculate the Total Revenue Generated by Events for Each User Using a Correlated Subquery.



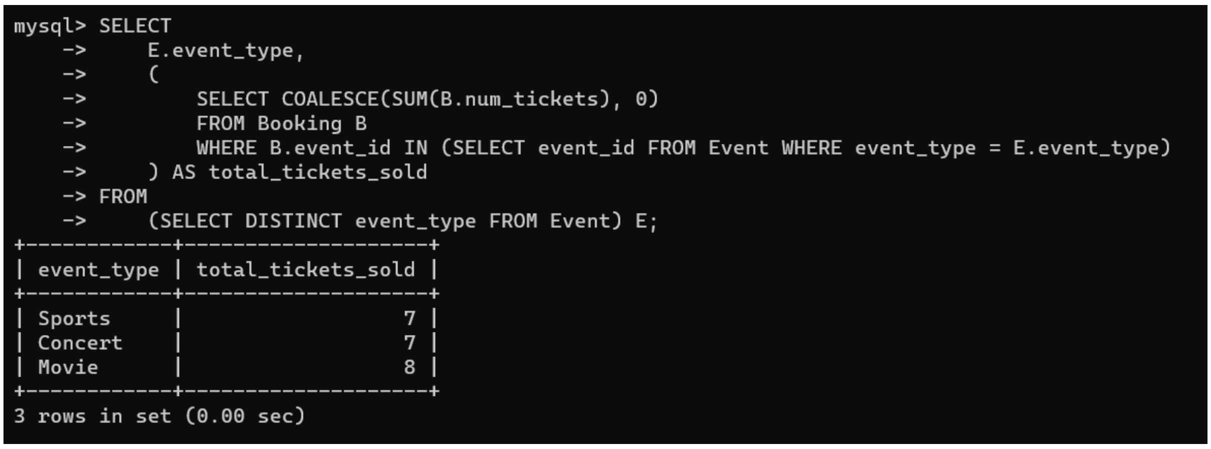
```
mysql> SELECT C.customer_id,C.customer_name,
    -> (SELECT COALESCE(SUM(B.total_cost), 0)
    -> FROM Booking B
    -> WHERE B.customer_id = C.customer_id
    -> ) AS total_revenue
    -> FROM Customer C;
+-------------+---------------+---------------+
| customer_id | customer_name | total_revenue |
+-------------+---------------+---------------+
|           1 | Amit Patel    |       3000.00 |
|           2 | Neha Sharma   |       1800.00 |
|           3 | Raj Singh     |        500.00 |
|           4 | Pooja Verma   |       1500.00 |
|           5 | Sandeep Kumar |        600.00 |
|           6 | Meera Kapoor  |        200.00 |
|           7 | Rahul Sharma  |       6000.00 |
|           8 | Neha Verma    |       1200.00 |
|           9 | Rajesh Singh  |       2400.00 |
|          10 | Anjali Gupta  |        800.00 |
+-------------+---------------+---------------+
10 rows in set (0.00 sec)
```

9. List Users Who Have Booked Tickets for Events in a Given Venue Using a Subquery in the WHERE Clause.



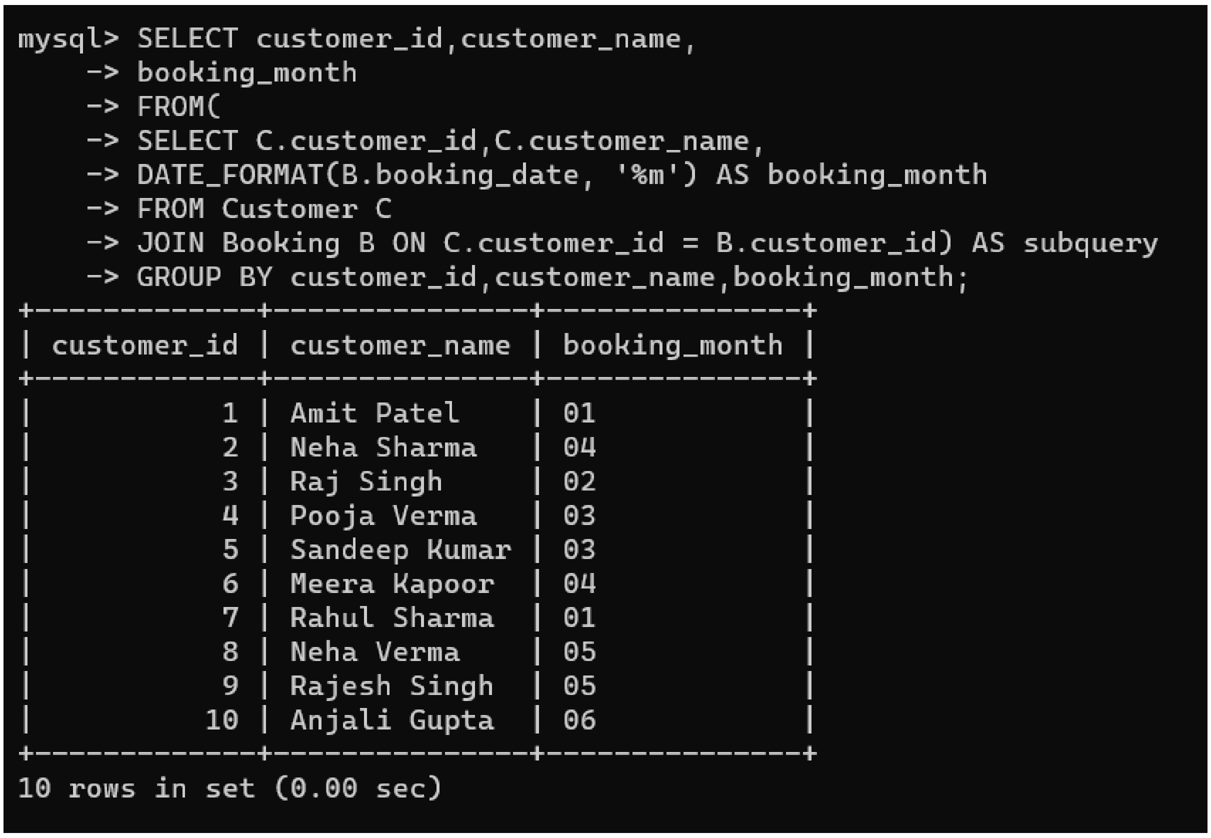
```
mysql> DELIMITER //
mysql>
mysql> CREATE PROCEDURE GetUserBookingsByVenue(IN p_venue_id INT)
    -> BEGIN
    -> SELECT C.customer_id,C.customer_name
    -> FROM Customer C
    -> WHERE EXISTS (
    -> SELECT 1
    -> FROM Booking B
    -> JOIN Event E ON B.event_id = E.event_id
    -> WHERE B.customer_id = C.customer_id
    -> AND E.venue_id = p_venue_id
    -> );
    -> END //
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> DELIMITER ;
mysql> CALL GetUserBookingsByVenue(2);
+-------------+---------------+
| customer_id | customer_name |
+-------------+---------------+
|           1 | Amit Patel    |
|           7 | Rahul Sharma  |
+-------------+---------------+
2 rows in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)
```

10. Calculate the Total Number of Tickets Sold for Each Event Category Using a Subquery with GROUP BY



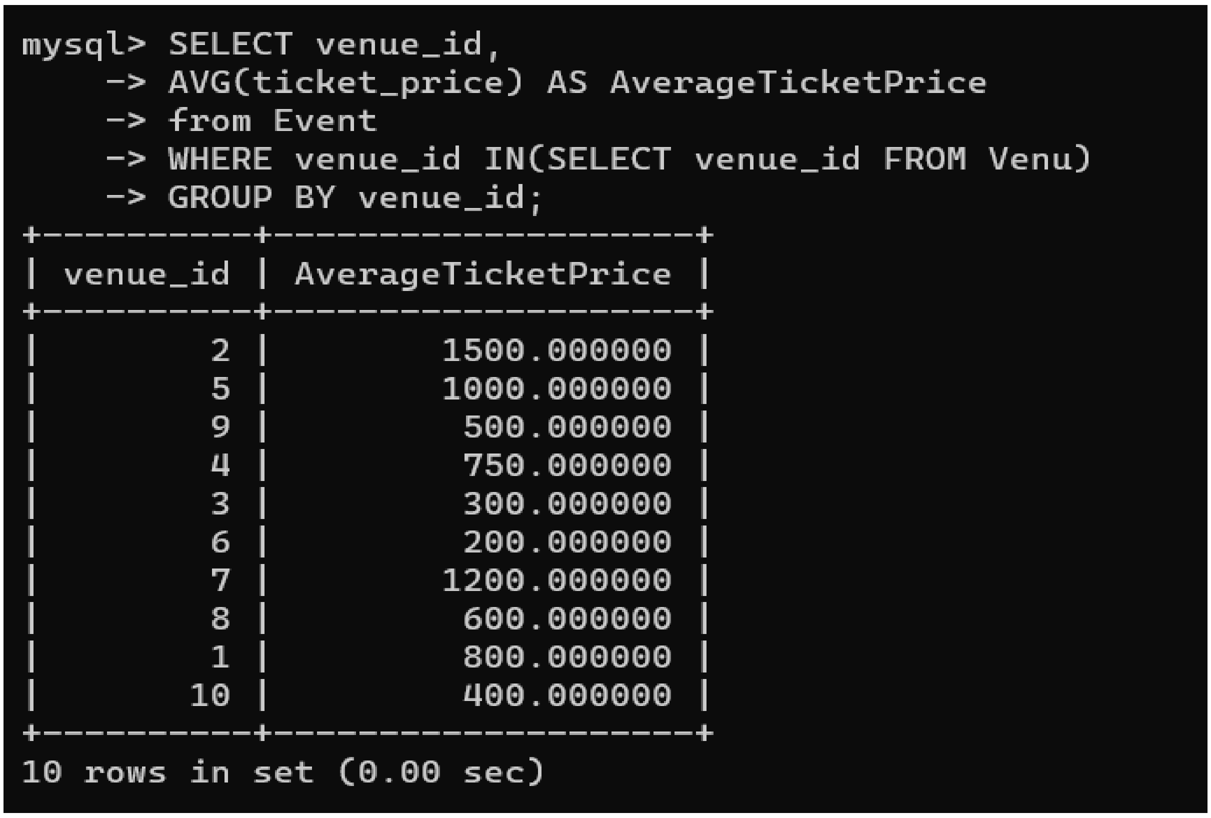
```
mysql> SELECT
    ->     E.event_type,
    ->     (
    ->         SELECT COALESCE(SUM(B.num_tickets), 0)
    ->         FROM Booking B
    ->         WHERE B.event_id IN (SELECT event_id FROM Event WHERE event_type = E.event_type)
    ->     ) AS total_tickets_sold
    -> FROM
    ->     (SELECT DISTINCT event_type FROM Event) E;
+------------+--------------------+
| event_type | total_tickets_sold |
+------------+--------------------+
| Sports     |                  7 |
| Concert    |                  7 |
| Movie      |                  8 |
+------------+--------------------+
3 rows in set (0.00 sec)
```

11. Find Users Who Have Booked Tickets for Events in each Month Using a Subquery with DATE_FORMAT



```
mysql> SELECT customer_id,customer_name,
    -> booking_month
    -> FROM(
    -> SELECT C.customer_id,C.customer_name,
    -> DATE_FORMAT(B.booking_date, '%m') AS booking_month
    -> FROM Customer C
    -> JOIN Booking B ON C.customer_id = B.customer_id) AS subquery
    -> GROUP BY customer_id,customer_name,booking_month;
+-------------+---------------+---------------+
| customer_id | customer_name | booking_month |
+-------------+---------------+---------------+
|           1 | Amit Patel    | 01            |
|           2 | Neha Sharma   | 04            |
|           3 | Raj Singh     | 02            |
|           4 | Pooja Verma   | 03            |
|           5 | Sandeep Kumar | 03            |
|           6 | Meera Kapoor  | 04            |
|           7 | Rahul Sharma  | 01            |
|           8 | Neha Verma    | 05            |
|           9 | Rajesh Singh  | 05            |
|          10 | Anjali Gupta  | 06            |
+-------------+---------------+---------------+
10 rows in set (0.00 sec)
```

12. Calculate the Average Ticket Price for Events in Each Venue Using a Subquery



```
mysql> SELECT venue_id,
    -> AVG(ticket_price) AS AverageTicketPrice
    -> from Event
    -> WHERE venue_id IN(SELECT venue_id FROM Venu)
    -> GROUP BY venue_id;
+----------+--------------------+
| venue_id | AverageTicketPrice |
+----------+--------------------+
|        2 |        1500.000000 |
|        5 |        1000.000000 |
|        9 |         500.000000 |
|        4 |         750.000000 |
|        3 |         300.000000 |
|        6 |         200.000000 |
|        7 |        1200.000000 |
|        8 |         600.000000 |
|        1 |         800.000000 |
|       10 |         400.000000 |
+----------+--------------------+
10 rows in set (0.00 sec)
```